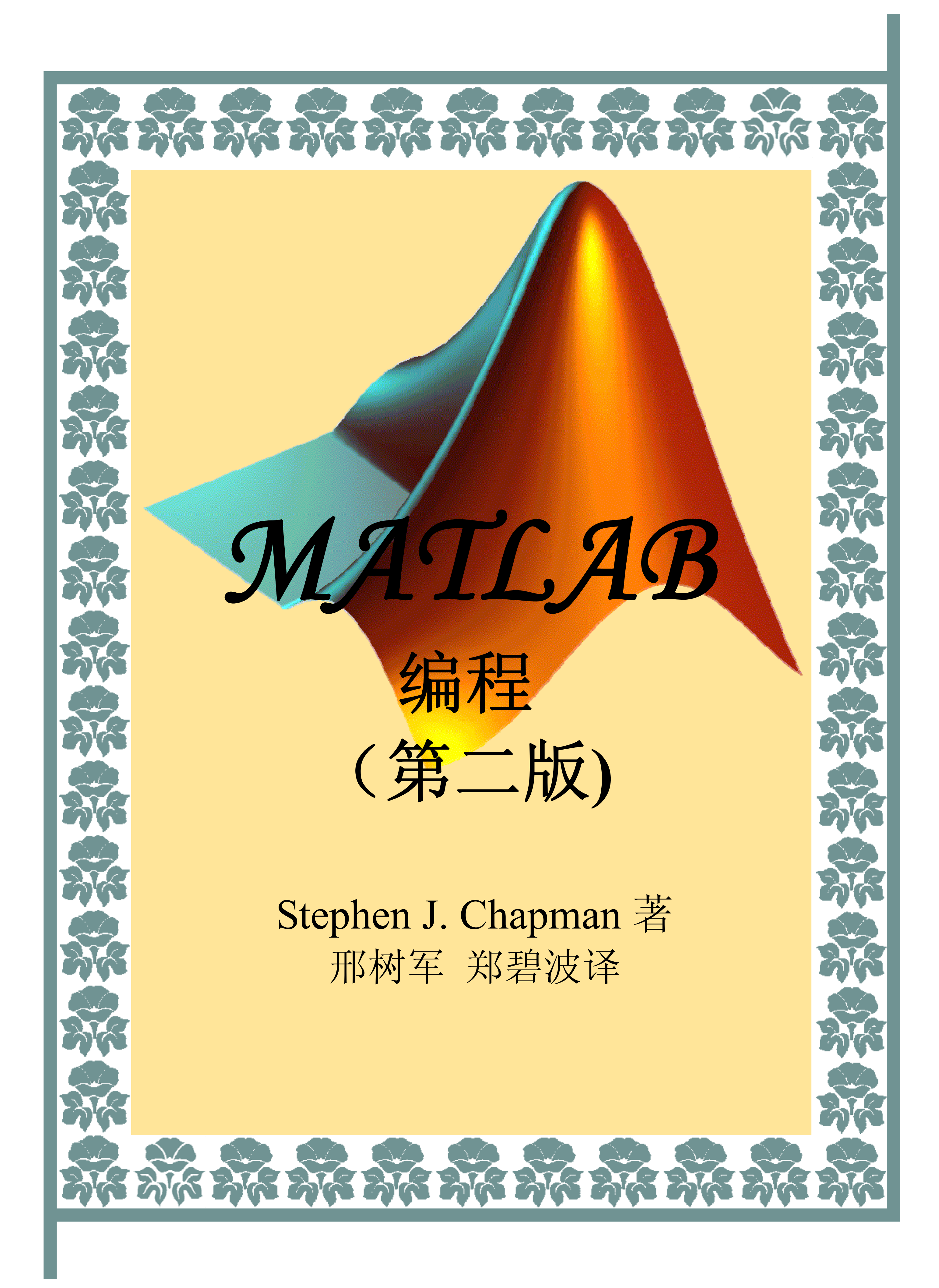


MATLAB 编程

（第二版)

Stephen J. Chapman 著

邢树军 郑碧波译

国外高校电子信息类优秀教材（英语影印版）

MATLAB 编程

（第二版）

MATLAB Programming for Engineers

（Second Edition）

Stephen J. Chapman 著

邢树军 郑碧波 译

前言

郑碧波

本文为 Stephen J.Chapman《**MATLAB** 编程（第二版）》英文影印版的中文译本，这本书对初学者很好的入门教材。之前从网上下载到邢树军兄弟译的 pdf 版本，邢的这个版本用起来有以下几方面不便：

1.文件较大。

2.打开之后总是提示播放某首歌，很烦。

3.源代码和图形采用了大量的图片，不方便学习者直接复制到 **MATLAB** 中进行调试学习。即使是文本的源代码，也经常出错，主要是每一行语句末尾的"；"为中文编码的分号。

4.个别字翻译错误（sorry，我的这个版本也没有完全消除），但我有发现的已经更正。

5.第十章翻译不完全。本来全书只有十章，在第十章还没有翻译结束时插进了第十一章。

6.附录 A 和 B 没有（翻译）提供。其中附录 B 为练习答案，其实这对初学者来说，是极其重要的。

于是就想着改进该版本的。本版本在邢译本的基础上做如下修改：

1.转换成 doc 格式，重新排版（其实也没怎么排版）。

2.更换封面，去掉歌曲。

3.所有源代码测试正确。

4.用了较多新的彩色截图代替原来的。

5.提供了附录 A 和附录 B。

6.删除了原来 pdf 版本的第 11 章（原著没有）。

本版本有如下缺点

1.由于本人截图用 **MATLAB** 是 7.0 和 6.5 版本，在 winxp 中运行。一些截图与原英文版本不一样，原因一是与原英文版本 **MATLAB** 6.0 不同，原因二是因于产生图形的数据无法找到相同的，好在基础知识方面的学习差别不大。

2.第十章前几节仍为树军兄弟翻译，后面部分由我翻译，但由于下载不到 6.0 版本，只好用 6.5 版本，但使用 6.5 版本会出现 GUI 编程代码不一样的情况，从忠于原著角度考虑感觉不爽，后来想想，既然网上都找不到 6.0 版本，那么使用 6.5 版本岂不更方便读者？！

3.附录 B 与前面的专有名词翻译可能会不一致。例如 cell array 一词，邢把它译为"单元阵列"，而我把它称为"胞数组"。我是一个初学者，不知权威的书是如何翻译的。不管如何，我已经做了相应说明，希望不会引起读者太大的不便。

本人英语也忘记得七七八八了。在翻译过程中肯定有不少不恰当的地方，如果你发现，请通知我，谢谢！

本书的绝大部分是邢树军翻译的，感谢邢树军兄弟的无私奉献，我仅仅是做些小小修补。其实翻译之后才发现，翻译过程是一项很繁琐很枯燥的事情！

郑碧波

（邮箱 bibofun@163.com）

2008-1-31

译序

邢树军

有这么一个故事：

在非洲草原上，有一种不起眼的动物叫吸血蝙蝠。它身体极小，却是野马的天敌。这种蝙蝠靠吸动物的血生存，它在攻击野马时，常附在马腿上，用锋利的牙齿极敏捷地刺破野马的腿，然后用尖尖的嘴吸血。无论野马怎样蹦跳、狂奔，都无法驱逐这种蝙蝠。蝙蝠却可以从容地吸附在野马身上，落在野马头上，直到吸饱吸足，才满意地飞去。而野马常常在暴怒、狂奔、流血中无可奈何地死去。

动物学家在分析这一问题时，一致认为吸血蝙蝠所吸的血量是微不足道的，远不会让野马死去，野马的死亡是它暴怒的习性和狂奔所致。

这个故事与 **MATLAB** 似乎没有任何的联系。编程初学者习惯于接到问题，就想在计算机编出好的程序来，而且要求没有错误。这是不可能的，时间一长就会像这只马一样变得暴怒，从而失去对编程的兴趣。所以，我翻译的这本书要求我们养成良好的编程习惯，强调自上而下的编程方法。希望读者能够理解这个方法。

自上而下的编程方法，不仅适用于编程，而且适用于我们做事。具体步骤为

（1）首先对你所面临的问题进行精确描述，即你遇到的到底是什么问题

（2）定义输入输出量，即我们的目的和现有的条件

（3）设计算法，逐层分解，逐步求精，这是做事谋的一个过程，大部分人做事出错就出在这个地方，懒得到逐层分解，逐步求精，总是妄想一步登天。

（4）把算法转化为 **MATLAB** 语言，这一步是踏踏实实做的过程，没有这一步你什么也做不成。

（5）检测程序，我们做的东西是否正确，不正确检测错误。这一步是需要耐心的，不要怕错误，每一次检测都会让你提高。做事时，不要因一些错误而畏缩不前。

在翻译的过程中，有很多的时侯都想放弃。有时也想不干了，但是每当这时，我就会找个人聊天，然后去思考，我遇到的问题是什么？是没有方法，还有心浮气躁。而多半的原因是因为后者，就像那匹马一样，因小事而丢了“性命”。于是，我翻译这本书，最大的收获是自上而下的编程方法。

序言

MATLAB(矩阵实验室的简称)是一种专业的计算机程序，用于工程科学的矩阵数学运算。但在以后的几年内，它逐渐发展为一种极其灵活的计算体系，用于解决各种重要的技术问题。**MATLAB** 程序执行 **MATLAB** 语言，并提供了一个极其广泛的预定义函数库，这样就使得技术工作变得简单高效。在解决工程技术问题方面，**MATLAB** 比其它任何计算机语言（包括 FORTAN 和 C）都简单高效。本书将介绍 **MATLAB** 语言，并向大家展示如何运用它去解决经典的技术问题。

本书把 **MATLAB** 当作一种技术编程语言来教，向学生展示如何编写简洁高效，有充分根据的程序。本书不主张把 **MATLAB** 数以百计的函数全部介绍给大家，而是教给学生怎样使用 **MATLAB** 语言，以及如何使用网上在线工具去加载你所需要的函数。本书的前六章是“编程设计介绍”的课程，专门为大一的工科学生而设计编写的。这部分内容以 9 周，每周 3 个小时教学为宜。其余的章节涵盖了一些高级的主题，例如输入输出接口（input/output interfaces）和图形用户界面（GUI）。这些章节可能需要更长时间的学习，也可以作为使用 **MATLAB** 的工科学生和有经验的工程师的参考。

第二版的变化

本书与 **MATLAB**6.0 和 6.1 相匹配，但基本的 **MATLAB** 语言与 **MATLAB** 5.0 大体相一致，而集成工具，窗口和帮助子系统都发生了巨大的变化.还有 **MATLAB** 6.0 许多新的完整的 **MATLAB** 图形用户界面.于是还在使用 6.0 以前版本的用户应注意到:在第十章的对于图形用户界面(GUI)的发展并不适用于他们.用 **MATLAB** 进行科技编程的优点 **MATLAB** 语言相对于传统的科技编程语言有诸多的优点。主要包括：

1．易用性

MATLAB 是种解释型语言，就像各种版本的 BASIC。和 BASIC 一样，它简单易用程序可用作便笺簿求打在命令行处表达式的值，也可执行预先写好的大型程序。在 **MATLAB** 集成开发环境下，程序可以方便的编写，修改和调试。这是因为这种语言极易使用，对于教育应用和快速建立新程序的原型，它是一个理想的工具。

许多的编程工具使得 **MATLAB** 十分简单易用。这些工具包括：一个集成的编译/调试器，在线文件手册，工作台和扩展范例。

2．平台独立性

MATLAB 支持许多的操作系统，提供了大量的平台独立的措施。在本书编写的时侯，windows 98/2000/NT 和许多版本的 UNIX 系统都支持它。在一个平台上编写的程序，在其它平台上一样可以正常运行，在一个平台上编写的数据文件在其它平台上一样可以编译。因此用户可以根据需要把 **MATLAB** 编写的程序移植到新平台。

3．预定义函数

MATLAB 带有一个极大的预定义函数库，它提供了许多已测试和打包过的基本工程问题的函数。例如，假设你正在编写一个程序，这个程序要求你必须计算与输入有关的统计量。在许多的语言中，你需要写出你所编数组的下标和执行计算所需要的函数，这些函数包括其

数学意义，中值，标准误差等。像这样成百上千的函数已经在 **MATLAB** 中编写好，所以让编程变得更加简单。

除了植入 **MATLAB** 基本语言中的大量函数，还有许多专用工具箱，以帮助用户解决在具体领域的复杂问题。例如，用户可以购买标准的工具箱以解决在信号处理，控制系统，通信，图象处理，神经网络和其他许多领域的问题。

4．机制独立的画图

与其他语言不同，**MATLAB** 有许多的画图和图象处理命令。当 **MATLAB** 运行时，这些标绘图和图片将会出现在这台电脑的图象输出设备中。此功能使得 **MATLAB** 成为一个形象化技术数据的卓越工具。

5．用户图形界面

MATLAB 允许程序员为他们的程序建立一个交互式的用户图形界面。利用 **MATLAB** 的这种功能，程序员可以设计出相对于无经验的用户可以操作的复杂的数据分析程序。

6．MATLAB 编译器

MATLAB 的灵活性和平台独立性是通过将 **MATLAB** 代码编译成设备独立的 P 代码，然后在运行时解释 P 代码来实现的。这种方法与微软的 VB 相类似。不幸的是，由于 **MATLAB** 是解释性语言，而不是编译型语言，产生的程序执行速度慢。当我们遇到执行速度慢的程序时，我们将会指出其这一特性。

本书的特点

本书诸多特点主要是向大家强调如何编写可靠的程序。这些特性不仅为初学 **MATLAB** 的学生服务，而且也为在工作岗位上的熟练者服务。

1．强调自上而下的编程方法

本书在第三章引入自上而下的编程方法，然后在以后的课程中坚持使用这种方法去解决问题。这种方法要求学生在开始编写代码前先做大体的设计。在其他工作开始之前，应强调问题解决方案的确定和输入输出量的定义。一旦一个问题被适当地确定了，我们会教给学生怎样逐步分解为一连串小的问题，然后执行这些小的问题就像执行独立的子程序或函数一样。最后我们将运用这种方法向大家介绍检测编程全过程的重要性，包括整个程序的单元检测和最终产品的彻底检测。

本书教授的编程流程归纳如下：

- 清晰地描述出你所要解决的问题。
- 定义出程序所要求的输入量和程序所应的输出量。
- 描述出你所要编写程序的算法，这一步将运用到自上而下和逐步分解的设计方法，用伪代码和流程图来描述。
- 把算法转化为 **MATLAB** 语句。
- 检测 **MATLAB** 程序。这一步包括单个函数的单元检测，也包括最终程序的详尽检测。

2. 强调使用函数

本书强调使用函数在逻辑上把大问题分解成小的子问题。它也强调在组合成最终的程序之前，先检测单元函数的重要性。本书也会介绍一些编程隐患和如何避免的方法。

3. 强调 MATLAB 工具的应用

本书将教会你适当使用 **MATLAB** 提供的工具，使编程和调试变得简单。这些工具包括：launch pad，编译调试器，工作台，帮助台和 GUI 设计工具。

4. 好的编程练习提示框

为了方便提示学生起见好的编程习惯被突出显示出来。每章的编程练习将会在章末做总结。示例如下：

好的编程习惯
为了增强程序的可读性，在 if 结构体开头缩进两格或更多的空间。

5. 编程隐患提示框

这些常见错误提示框突出显示一些常见的编程错误，故初学者能够参考，以致不出类似的错误。示例如下：

编程隐患
确保变量名的前三十一个字符是独一无二的，否则，MATLAB 将辨认不出两变量的不同。

6. 强调数据结构

第七章向大家详细地介绍 **MATLAB** 数据结构，数据结构包括稀疏阵列，单元阵列和结构阵列.这一章通过用户掌握图解和用户图形接口来向大家介绍如何适当使用这些数据结构.

授课特点

本课程的前六个章节是作为工程学一年级学生的《编程及问题解决入门》课程的教材而书写的.它将耗费九周的时间，每周三个小时.如果时间不充足的话，第六章可以删除不讲.前五章仍是编程基础和应用 **MATLAB** 解决问题，这就要求我们工程学教学工作者.其余的章节涵盖了 **MATLAB** 的高级内容，这些内容将在学生以后工作中将会非常的有用。它包括高级输入输出和用户图形界面的设计。

本书许多的特性可帮助学生理解。总共有 15 个小测试零散的分布在全书中，其答案存在于附录 B 中。这些小测试将有助于自我理解。此外，大约还有 140 道章末课后题。一部分精选的课后题的答案可在本书的网站上找到，当然全部的答案可在教师的指导书中找到好的编程习惯在所有的章节中被突出显示出来，常见错误提示框突出显示一些常见的编程错

误。章末的材料包括好的编程习惯的总结和 **MATLAB** 命令和函数的总结。与本书相配套的教师参考手册包含所有章末练习的答案。本书所有例子的代码都可在本书的网站上得到，而章末练习的答案的代码只能在手册中得到。

对使用者最后的提示

不论我多么努力的校对本书的文本，印刷错误总是难免的。如果你发现了一些错误，你可以通过出版商通知我，我将在再版的时候做到最好。非常感谢你们在这方面的帮助。我列举了一系列的错误和更正在网站 http://info.brookscole.com/chapman。欢迎访问。

鸣谢

我应当感谢 bill stenquist 和他在 books/cole 公司的同仁们对于本书的支持，有了他们的支持才有本书的出版.对于本书第一版的反馈我深感幸慰.这是我们通力合作的结果.

我还应感谢我的妻子 ROSE 和我的孩子们 Avi，David，Rachel，Aaron，Sarah，Naomi，Shira，and Devorah.他们都是乐观的人，在工作上给于我很大的鼓励。

斯蒂芬 *J.* 查普曼

目录

第一章 MATLAB 介绍 1
1.1 MATLAB 的优点 1
1. 易用性 1
2. 平台独立性 1
3. 预定义函数 1
4. 机制独立的画图 2
5. 用户图形界面 2
6. MATLAB 编译器 2
1.2 MATLAB 的缺点 2
1.3 MATLAB 的开发环境 2
1.3.1 MATLAB 桌面 2
1.3.2 命令窗口（TheCommandWindow） 3
1.3.3 历史命令窗口（The History Command Window） 4
1.3.4 启动平台（the launch pad） 5
1.3.5 编辑调试器 5
1.3.6 图像窗口（Figure Windows） 6
1.3.7 MATLAB 工作区 6
1.3.8 工作区浏览器 7
1.3.9 MATLAB 帮助 8
1.3.10 一些重要的命令 9
1.3.11 MATLAB 搜索路径 10
1.4 把 MATLAB 当作便笺薄来使用 11
测试 1.1 12
1.5 总结 12
1.5.1 MATLAB 总结 13
1.6 练习 13
第二章 MATLAB 基础 15
2.1 变量和数组 15
2.2 MATLAB 变量的初始化 17
2.2.1 用赋值语句初始化变量 17
2.2.2 用捷径表达式赋值 19
2.2.3 用内置函数来初始化 19
2.2.4 用关键字 input 初始化变量 20
测试 2.1 20
2.3 多维数组 21
2.3.1 多维数组在内存中的存储 22
2.3.1 用单个下标访问多标数组 22
2.4 子数组 23
2.4.1 end 函数 23
2.4.2 子数组在左边的赋值语句的使用 24
2.4.3 用一标量来给子数组赋值 25
2.5 特殊变量 25
测试 2.2 26
2.6 显示输出数据 26
2.6.1 改变默认格式 27

2.6.2 disp 函数........27
2.6.3 用 fprintf 函数格式化输出数据........28
2.7 数据文件........28
测试 2.3........29
2.8 标量运算和数组运算........30
2.8.1 标量运算符........30
2.8.2 数组运算和矩阵运算........30
例 2.1........32
2.9 运算的优先级........33
例 2.2........33
测试 2.4........34
2.10 MATLAB 的内建函数........34
2.10.1 选择性结果........34
2.10.2 带数组输入的 MATLAB 函数的应用........34
2.10.3 常见的 MATLAB 函数........35
2.11 画图入门........35
2.11.1 简单的 xy 画图........36
2.11.2 打印图象........37
2.11.3 联合作图........37
2.11.4 线的颜色,线的形式,符号形式和图例........38
2.11.5 对数尺度........40
2.12 例子........41
例 2.3........41
例 2.4........42
例 2.5........44
2.13 调试 MATLAB 程序........46
2.14 总结........47
2.14.1 好的编程习惯总结........47
2.14.2 MATLAB 总结........48
2.15 练习........50
第三章 分支语句和编程设计........53
3.1 自上而下的编程方法简介........53
3.2 伪代码的应用........56
3.3 关系运算符和逻辑运算符........56
3.3.1 关系运算符........56
3.3.2 小心==和~=运算符........57
3.3.3 逻辑运算符........58
例 3.1........59
3.3.4 逻辑函数........59
测试 3.1........60
3.4 选择结构(分支语句)........60
3.4.1 if 结构........60
3.4.2 if 结构举例........62
例 3.2........62
例 3.3........64
3.4.3 关于 if 结构使用的注意事项........66
例 3.4........67
3.4.4 switch 结构........68
3.4.5 try/catch 结构的应用........69
测试 3.2........70

3.5 附加的画图特性......70
3.5.1 控制 x，y 轴绘图的上下限......70
3.5.2 在同一坐标系内画出多个图象......73
3.5.3 创建多个图象......73
3.5.4 子图象......74
3.5.5 对画线的增强控制......75
3.5.6 文本字符串的高级控制......76
3.5.7 极坐标图象......77
例 3.5......77
例 3.6......79
例 3.7......80
3.5.8 注释并保存图象......82
测试 3.3......84
3.6 程序调试的进一步说明......84
3.7 总结......87
3.7.1 好的编程习惯的总结......88
3.7.2 MATLAB 总结......88
3.8 练习......88
第四章 循环结构......91
4.1 while 循环......91
例 4.1......91
4.2 for 循环......95
例 4.2......96
例 4.3......96
例 4.4......99
4.2.1 运算的细节......100
例 4.5......101
4.2.2 break 和 continue 语句......102
4.2.3 循环嵌套......103
4.3 逻辑数组与向量化......104
4.3.1 逻辑数组的重要性......105
例 4.6......106
4.3.2 用 if/else 结构和逻辑数组创建等式......107
测试 4.1......108
4.4 附加例子......109
例 4.7......109
例 4.8......114
4.5 总结......120
4.5.1 好的编程习惯总结......120
4.5.2 MATLAB 总结......120
4.6 练习......120
第五章 自定义函数......127
5.1 MATLAB 函数简介......128
5.2 在 MATLAB 中传递变量：按值传递机制......132
例 5.2 数据排序......136
5.3 选择性参数......140
例 5.3 选择性参数的应用......141
测试 5.1......142
5.4 用全局内存分享数据......143
5.5 在函数调用两次之间本地数据的存储......148

5.6 函数的函数(function functions)，......152
5.7 子函数和私有函数......155
5.8 总结......156
5.9 练习......156
第六章 复数数据、字符数据和附加画图类型......165
6.1 复数数据......165
6.1.1 复变量（complex variables）......166
6.1.2 带有关系运算符的复数的应用......166
6.1.3 复函数（complex function）......166
1. 类型转换函数......167
2. 绝对值和幅角函数......167
3. 数学函数......167
例 6.1......167
6.1.4 复数数据的作图......169
6.2 字符串函数（string functions）......173
6.2.1 字符转换函数......173
6.2.2 创建二维字符数组......174
6.2.3 字符串的连接......174
6.2.4 字符串的比较......175
6.2.5 在一个字符串中查找/替换字符......176
6.2.6 大小写转换......177
6.2.7 字符串转换为数字......178
6.2.8 数字转化为字符串......178
6.2.9 总结......179
例 6.2......180
6.3 多维数组......183
6.4 关于二维作图的补充说明......185
6.4.1 二维作图的附加类型......185
6.4.2 作图函数......189
6.4.3 柱状图......190
6.5 三维作图......191
6.5.1 三维曲线作图......191
6.5.2 三维表面，网格，等高线图象......193
6.6 总结......196
6.6.1 好的编程习惯总结......196
6.6.2 MATLAB 函数与命令总结......196
6.7 练习......197
第七章 稀疏矩阵 单元阵列 结构......199
7.1 稀疏矩阵......199
7.1.1 sparse 数据类型......200
例 7.1......202
7.2 单元阵列(cell array)......204
7.2.1 创建单元阵列......205
7.2.2 单元创建者——大括号({})的应用......206
7.2.3 查看单元阵列的内容......206
7.2.4 对单元阵列进行扩展......207
7.2.5 删除阵列中的元素......208
7.2.6 单元阵列数据的应用......208
7.2.7 字符串单元阵列......209
7.2.8 单元阵列的重要性......209

7.2.9 单元阵列函数总结......212
7.3 结构数组......212
7.3.2 增加域到结构......214
7.3.3 删除结构中的域......214
7.3.4 结构数组中数组的应用......215
7.3.5 函数 getfield 和函数 setfield......216
7.3.6 对结构数组应用 size 函数......217
7.3.7 结构的嵌套......217
7.3.8 struct 函数总结......218
测试 7.1......218
7.4 总结......219
7.4.1 好的编程习惯总结......219
7.4.2 MATLAB 函数命令总结......219
7.5 练习......220
第八章 输入/输出函数......221
8.1 函数 textread......221
8.2 关于 load 和 save 命令的进一步说明......222
8.3 MATLAB 文件过程简介......223
8.4 文件的打开与关闭......224
8.4.1 fopen 函数......224
8.4.2 fclose 函数......226
8.5 二进制 I/O 函数......226
8.5.1 fwrite 函数......226
8.5.2 fread 函数......227
例 8.1 读写二进制数据......228
测试 8.1......229
8.6 格式化 I/O 函数......229
8.6.1 fprint 函数......229
8.6.2 格式转换指定符的理解......231
8.6.3 如何使用格式字符串......232
例 8.2 产生一个信息表......233
8.6.4 fscanf 函数......234
8.6.5 fgetl 函数......235
8.6.6 fgets 函数......235
8.7 格式化和二进制 I/O 函数的比较......236
例 8.3 格式化和二进制 I/O 文件的比较......236
测试 8.2......239
8.8 文件位置和状态函数......239
8.8.1 exist 函数......239
例 8.4 打开一个输出文件......240
8.8.2 函数 ferror......241
8.8.3 函数 foef......241
8.8.4 函数 ftell......242
8.8.5 函数 frewind......242
8.8.6 函数 fseek......242
例 8.5......242
8.9 函数 uiimport......246
8.10 总结......248
8.10.1 好的编程习惯总结......248
8.10.2 MATLAB 总结......248

8.11 练习......249

第九章 句柄图形......251

9.1 MATLAB 图形系统......251
9.2 对象句柄......252
9.3 对象属性的检测和更......252
9.3.1 在创建对象时改变对象的属性......252
9.3.2 对象创建后改变对象的属性......252
例 9.1......256
9.4 用 set 函数列出可能属性值......259
9.5 自定义数据......259
9.6 对象查找......260
9.7 用鼠标选择对象......261
例 9.2......262
9.8 位置和单位......264
9.8.1 图象(figure)对象的位置......264
9.8.2 坐标系对象和 uicontrol 对象的位置......265
9.8.3 文本(text)对象的位置......265
例 9.3......265
9.9 打印位置......268
9.10 默认和 factory 属性......268
9.11 图形对象属性......269
9.12 总结......269
9.13 练习......270

第十章 用户图形界面......271

10.1 用户界形界面是如何工作的......271
10.2 创建并显示用记图形界面......271
10.2.1 盖头下的一瞥......279
10.2.2 一个响应子函数的结构......281
10.2.3 给图象增加应用程序数据......281
10.2.4 一些有用的函数......282
10.3 对象属性......282
10.4 图形用户界面组件......283
10.4.1 文本域(Text Fields)......284
10.4.2 编辑框(Edit Boxes)......284
10.4.3 框架(Frames)......285
10.4.4 按钮(Pushbuttons)......285
10.4.5 开关按钮(Toggle Buttons)......285
10.4.6 复选和单选按钮(Checkboxes and Radio Buttons)......286
10.4.7 下拉菜单(Popup Menus)......288
10.4.8 列表框(List Boxes)......289
10.4.9 滑动条(Sliders)......291
例 10.1......292
10.5 对话框......294
10.5.1 错误和警告对话框......294
10.5.2 输入对话框......295
10.5.3 打开与保存对话框......295
10.6 菜单......296
10.6.1 禁用默认菜单......298
10.6.2 创建自定义菜单......299
10.6.3 加速键与键盘助记键......299

10.6.4 创建上下文菜单......300
例 10.2 绘制数据点......300
测试 10.1......304
10.7 创建高效 GUIs 的技巧......304
10.7.1 工具提示......305
10.7.2 伪代码（p 码，pcode）......305
10.7.3 附加提高......305
例 10.3......306
10.8 总结......309
10.8.1 好的编程习惯总结......310
10.8.2 MATLAB 总结......310
10.9 练习......310

附录 A ASCII 字符集......313

附录 B 测试答案......314

测试 1.1......314
测试 2.1......315
测试 2.2......315
测试 2.3......315
测试 2.4......316
测试 3.1......316
测试 3.2......317
测试 3.3......317
测试 4.1......318
测试 5.1......318
测试 6.1......319
测试 7.1......319
测试 8.1......320
测试 8.2......321
测试 10.1......322

第一章 MATLAB 介绍

MATLAB(矩阵实验室的简称)是一种专业的计算机程序，用于工程科学的矩阵数学运算。但在以后的几年内，它逐渐发展为一种极其灵活的计算体系，用于解决各种重要的技术问题。**MATLAB** 程序执行 **MATLAB** 语言，并提供了一个极其广泛的预定义函数库，这样就使得技术工作变得简单高效。本书将介绍 **MATLAB** 语言，并向大家展示如何运用它去解决经典的技术问题。

MATLAB 是一个庞大的程序，拥有难以置信的各种丰富的函数；即使基本版本的 **MATLAB** 语言拥有的函数也比其他的工程编程语言要丰富的多。基本的 **MATLAB** 语言已经拥有了超过 1000 多个函数，而它的工具包带有更多的函数，由此扩展了它在许多专业领域的能力。本书无意将 **MATLAB** 的所有函数介绍给大家，而是让大家掌握编写调试和优化程序的基本功，还有一些重要函数的子集。所以从大量可利用的函数中筛选出你所需要的函数就显得尤为重要。

1.1 MATLAB 的优点

MATLAB 语言相对于传统的科技编程语言有诸多的优点。主要包括：

1. 易用性

MATLAB 是种解释型语言，就像各种版本的 BASIC。和 BASIC 一样，它简单易用程序可用作便笺簿求打在命令行处表达式的值，也可执行预先写好的大型程序。在 **MATLAB** 集成开发环境下，程序可以方便的编写，修改和调试。这是因为这种语言极易使用，对于教育应用和快速建立新程序的原型，它是一个理想的工具。

许多的编程工具使得 **MATLAB** 十分简单易用。这些工具包括：一个集成的编译/调试器，在线文件手册，工作台和扩展范例。

2. 平台独立性

MATLAB 支持许多的操作系统，提供了大量的平台独立的措施。在本书编写的时候，windows98/2000/NT 和许多版本的 UNIX 系统都支持它。在一个平台上编写的程序，在其它平台上一样可以正常运行，在一个平台上编写的数据文件在其它平台上一样可以编译。因此用户可以根据需要把 **MATLAB** 编写的程序移植到新平台。

3. 预定义函数

MATLAB 带有一个极大的预定义函数库，它提供了许多已测试和打包过的基本工程问题的函数。例如，假设你正在编写一个程序，这个程序要求你必须计算与输入有关的统计量。在许多的语言中，你需要写出你所编数组的下标和执行计算所需要的函数，这些函数包括其

数学意义，中值，标准误差等。像这样成百上千的函数已经在 **MATLAB** 中编写好，所以让编程变得更加简单。

除了植入 **MATLAB** 基本语言中的大量函数，还有许多专用工具箱，以帮助用户解决在具体领域的复杂问题。例如，用户可以购买标准的工具箱以解决在信号处理，控制系统，通信，图象处理，神经网络和其他许多领域的问题。

4. 机制独立的画图

与其他语言不同，**MATLAB** 有许多的画图和图象处理命令。当 **MATLAB** 运行时，这些标绘图和图片将会出现在这台电脑的图象输出设备中。此功能使得 **MATLAB** 成为一个形象化技术数据的卓越工具。

5. 用户图形界面

MATLAB 允许程序员为他们的程序建立一个交互式的用户图形界面。利用 **MATLAB** 的这种功能，程序员可以设计出相对于无经验的用户可以操作的复杂的数据分析程序。

6. MATLAB 编译器

MATLAB 的灵活性和平台独立性是通过将 **MATLAB** 代码编译成设备独立的 P 代码，然后在运行时解释 P 代码来实现的。这种方法与微软的 VB 相类似。不幸的是，由于 **MATLAB** 是解释性语言，而不是编译型语言，产生的程序执行速度慢。当我们遇到执行速度慢的程序时，我们将会指出其这一特性。

1.2 MATLAB 的缺点

MATLAB 有两个基本的缺点。

第一，它是解释型语言，其执行速度要比编译型语言慢得多。这个问题可以通过合理的 **MATLAB** 结构得到缓解，也可以在发行广泛使用前编译出 **MATLAB** 程序。

第二，他的费用较高。一个完全版 **MATLAB** 编译器的大小是一个 C 语言或 Fortan 语言编译器的 5 到 10 倍。但 **MATLAB** 能够节省大量的时间在科技编程方面，故 **MATLAB** 在商业编程过程中是节省成本的。尽管如此，相对于大多数考虑购买的人还是很昂贵的。幸运的是，它有一个价格便宜的学生专用版本，对学生来说它是学习 **MATLAB** 语言的一个重要工具。学生版的 **MATLAB** 和完全版的 **MATLAB** 是基本一致的。

1.3 MATLAB 的开发环境

1.3.1 MATLAB 桌面

任何一个 **MATLAB** 程序的基本组成单元是数组。数组是一组数据值的集合，这些数据

被编上行号和列号，拥有唯一的名称。数组中的单个数据是可以通过带有小括号的数组名访问，括号内有这个数据的行标和列标，中间用逗号隔开。标量也被 **MATLAB** 当作数组，只不过只有一行和一列。在第二章我们将学习如何创建和操作 **MATLAB** 数组。

当 **MATLAB** 运行时，有多种类型的窗口，有的用于接收命令，有的用于显示信息。三个重要的窗口有命令窗口；图像窗口；编辑/调试窗口；它们的作用分别为输入命令；显示图形；充许使用者创建和修改 **MATLAB** 程序。在本节课中我们将会看到这三个窗口的例子。

当 **MATLAB** 程序启动时，一个叫做 **MATLAB** 桌面的窗口出现了。默认的 **MATLAB** 桌面结构如图 1.1 所示。在 **MATLAB** 集成开发环境下，它集成了管理文件、变量和应用程序的许多编程工具。

在 **MATLAB** 桌面上可以得到和访问的窗口主要有：

- 命令窗口（The Command Window）
- 命令历史窗口（The Command History Window）
- 启动平台（Launch Pad）
- 编辑调试窗口（The Edit/Debug Window）
- 工作台窗口和数组编辑器（Workspace Browser and Array Editor）
- 帮助空间窗口（Help Browser）
- 当前路径窗口（Current Directory Browser）

我们将在本章的最后一节讨论这些工具的函数。

1.3.2 命令窗口（TheCommandWindow）

MATLAB 桌面的右边是命令窗口。在命令窗口中，用户可以在命令行提示符(>>)后输入一系列的命令，这些命令的执行也是在这个窗口中实现的。

举一个简单的例子，假设你要计算一个半径为 2.5m 的圆的面积。在命令窗口中的操作如下：

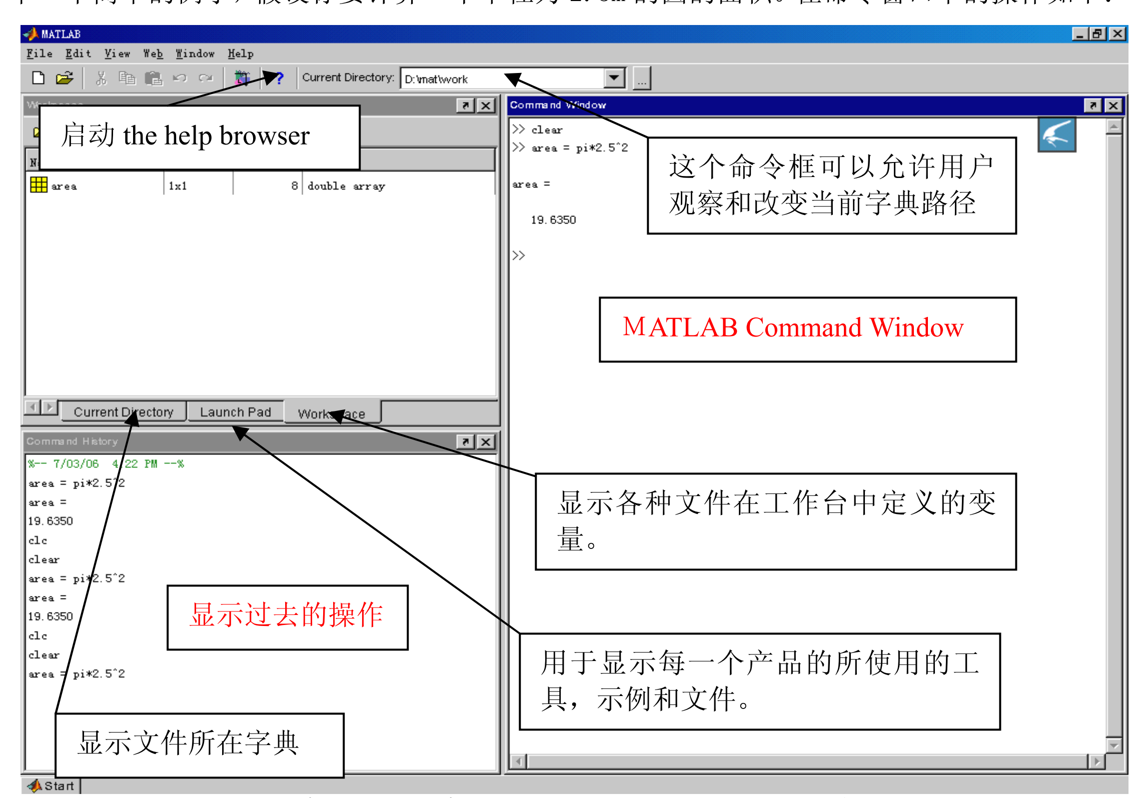


图 1.1MATLAB 桌面，具体桌面布局可能因机器的不同而会有的轻微的变化

```
>> area=pi*2.5^2
area =
```

```
19.6350
```

当回车键敲下的一瞬间，结果被计算了出来，并被存储到一个叫 area 的变量中（其实是一个 1×1 的数组)。这个变量的数值将显示在命令窗口(TheCommandWindows),如图 1.2 所示，而且这个变量能进行进一步的计算。(注意 π 是 **MATLAB** 预先定义好的变量，所以 pi 不需要预先声明。)

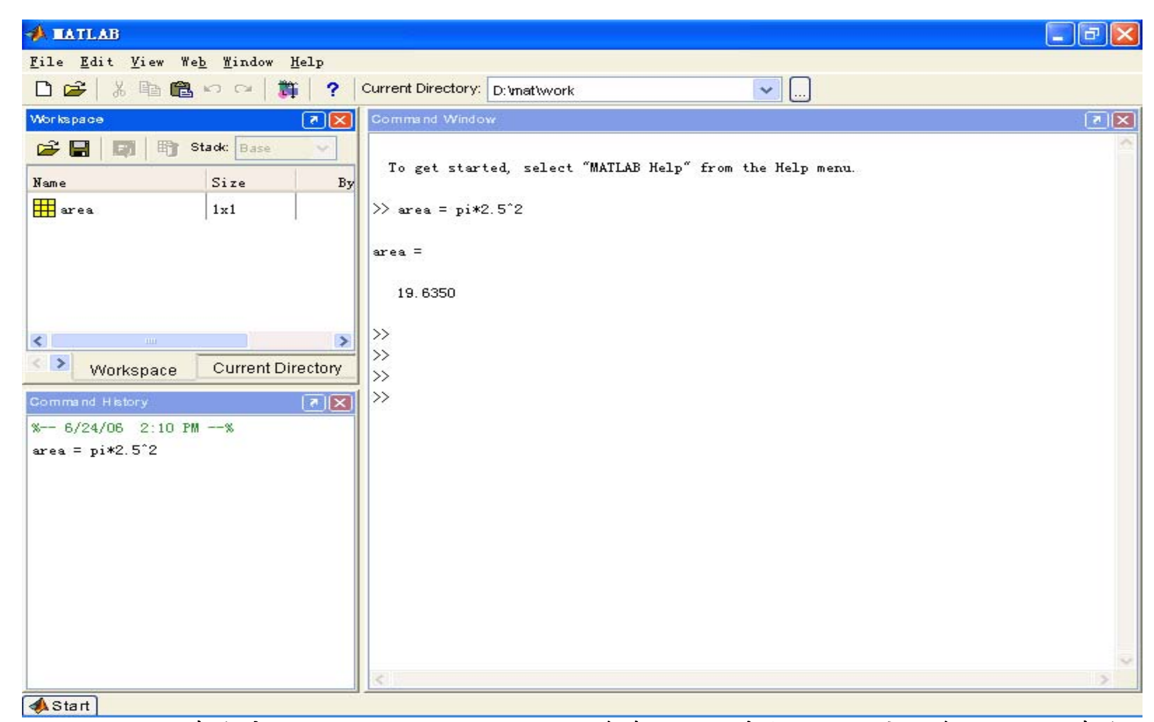


Figure1.2 命令窗口(The Command Windows)在桌面的右半部分。用户可在这里输入命令。

如果一个语句在一行内书写太长了，可能要另起一行接着写，在这种情况下我们需要在第一行末打上半个省略号（...），再开始第二行的书写。

举例如下，下面这两语句是等价的。

```
x1=1+1/2+1/3+1/4+1/5+1/6;
```

And

```
x1=1+1/2+1/3+1/4 ...
+1/5+1/6;
```

将一系列命令写入一个文件，在命令窗口(The Command Windows)输入此文件的文件名，然后 **MATLAB** 就开始执行这个文件，而不是用直接在命令窗口(The Command Windows)键入的方法，这样的文件叫做脚本文件(Script files),由于脚本文件(Script files)的扩展名为“.m”，所以这它也叫做 M 文件。

1.3.3 历史命令窗口（The History Command Window）

历史命令窗口（The History CommandWindow）用于记录用户在命令窗口(The Command Windows)，其顺序是按逆序排列的。即最早的命令在排在最下面，最后的命令排在最上面。这些命令会一直存在下去，直到它被人为删除。双击这些命令可使它再次执行。在历史命令窗口(The Command Windows)删除一个或多个命令，可以先选择，然后单击右键，这时就有一个弹出菜单出现，选择 Delete Section。任务就完成了。

1.3.4 启动平台（the launch pad）

启动平台是一个特殊的工具，为 **MATLAB** 和其工具箱提供帮助、demos、其他相关文件和应用程序等参考资料。这些信息是产品附带的，所有的参考资料都在每个产品或工具箱后面列出。不同的人拥有不同的产品，所以这个内容随个人安装的不同而不同。

Figure1.4 显示的是只带有基本 **MATLAB** 产品的参考资料的启动平台。双击其中一个条目，你将会得到 **MATLAB** 的帮助，运行 **MATLAB** 的示例，访问这个程序所支持的标准工具，或访问 **MATLAB** 在互联网上的网站。

1.3.5 编辑调试器

编辑调试器一般用于创建 M 文件，或者修改已存在的 M 文件。当你打开或修改一个 M 文件，编辑调试器会自动被调用。创建一个 M 文件的方法：

一、在菜单按“File/New/M-file”创建；

二、单击图标。

打开一个已存在的 M 文件也有两个方法：

一、按路径“File/Open”打开；

二、单击图标。

图 1.5 显示了一个包含有 M 文件的简单的编辑窗口

编程调试器是个重要的程序的文档编辑器，**MATLAB** 语言的一些特性会被不同的颜色表现出来。M 文件中的评论用绿色表示，变量和数字用黑色来表示，字符变量用红色表示，语言的关键字用蓝色表示。如图 1.5 显示了一个包含有 M 文件的简单的编辑窗口。这个文件是为了计算半径已知的圆的面积并输出结果。

```
% this m-file calculates the area of a circle,
% and display the result
radius=2.5;
area=pi*2.5^2;
string=['the area of the circle is ' , num2str(area)];
disp(string);
```

当 M 文件保存完后，在命令窗口(The Command Windows)中输入这个 M 文件的名字，它就可以被执行了。图 1.5 的输出结果为

```
>>calc_area
The area of the circle is 19.635
```

这个编辑器同样是个调试器，我们将会在第二章介绍它在调试方面的应用。

1.3.6 图像窗口（Figure Windows）

图像窗口主要是用于显示 **MATLAB** 图像。它所显示的图像可以是数据的二维或三维坐标图，图片，或用户图形接口。下面是一个简单的脚本文件(Script files)用于计算函数 sinx 并打印出图象。

```
% this m-file calculates and plots the
% function sin(x) for 0<=x<=6.
x=0:0.1:6;
y=sin(x);
plot(x,y);
```

如果此文件以 sin_x.m 为文件名保存，那么你可以在命令窗口(The Command Windows)输入此文件名就可以执行文件了。当脚本文件(Script files)被编译后，**MATLAB** 将会找开一个图象窗口，并在窗口打印出函数 sinx 的图象。

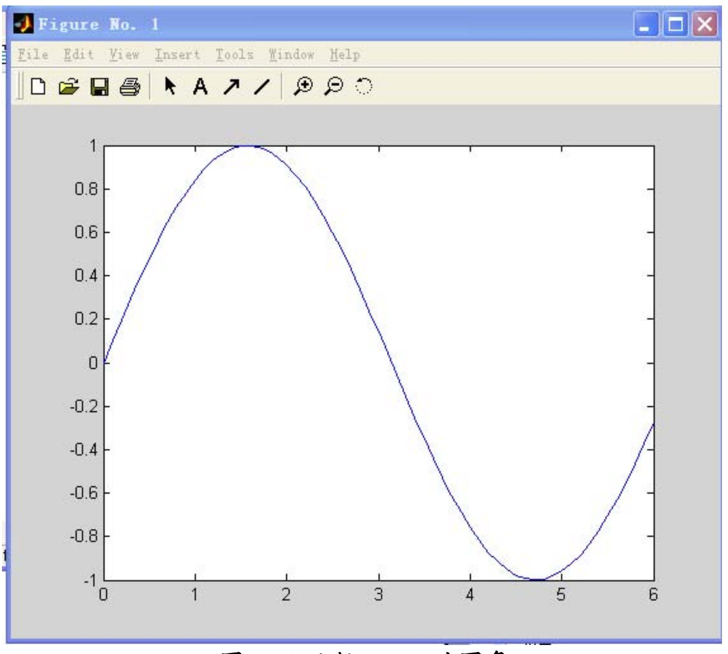


图 1.6 函数 sinx 的图象

1.3.7 MATLAB 工作区

像 z=10 这样的语句创建了一个变量 z，把 10 存储在其内，它保存在计算机的一段内存中，就是我们所常说的工作区。当一个专门的命令，M 文件或函数运行时，工作区是 **MATLAB** 所需要的所有变量和数组的集合。所有在命令窗口(The Command Windows)中执行的命令，和所有在命令窗口(The Command Windows)执行的脚本文件(Script files)都会被分配一个普通的分配空间，所以它们能共享变量。**MATLAB** 函数的拥有独立的工作区，这是函数区别于脚本文件(Script files)的一个重要特征。在后面的介绍我们将会看到的。

用 whos 命令将会产生一个在当前工作区内的所有变量和数组状况表。就以 M 文件 calc_area 和 sin_x 为说明，当两文件执行后，这个工作区所包含的变量有：

```
>> whos
  Name         Size                    Bytes  Class

  area         1x1                         8  double array
```

```
    radius        1x1                    8   double array
    string        1x32                  64   char array
    x             1x61                 488   double array
    y             1x61                 488   double array

Grand total is 156 elements using 1056 bytes
```

脚本文件(Script files)calc_area 创造了变量 area,radius 和 string，sin_x 创造了变量 x 和 y。请注意所有的变量在同一工作区，所以两个脚本文件(Script files)按顺序执行,第二个脚本文件(Script files)可以利用第一个脚本文件(Script files)所创建的变量。每一个变量和数组的内容可以通过在命令窗口(The Command Windows)中输入对应名字得到显示。例如 string 的内容如下:

```
>> string
string =
the area of the circle is 19.635
```

可用 clear 命令删除在本工作区的变量，格式如下

clear var1,var2 ...

var1，var2 是要删除变量的变量名。clear variables 命令或 clear 命令用于清除当前工作区中的所有变量。

1.3.8 工作区浏览器

当前工作区的内容也可以通过基于 GUI 的工作空间窗口检测到。工作空间窗口默认出现在 **MATLAB** 桌面的左上角，它提供了和 whos 命令可得到的相同的信息，并当工作区内的内容发生改变时，其内的信息也会随之更新。工作空间窗口（The workspace browser）允许用户改变工作区内的任何一个变量的内容。

典型的工作空间窗口(The workspace browser)如图 1.7.你能看到它显示的信息和 whos 命令得到的信息是一样的。双击这个窗口任一变量便产生了一个数组编辑器，这个编辑器允许用户修改保存在变量中的信息。

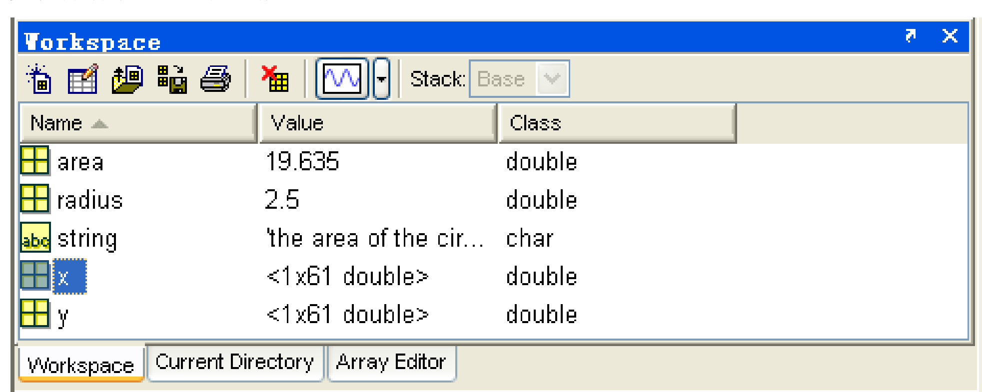


图 1.7 工作空间窗口（The workspace browser）

一个或多个变量可在工作空间内删除，先选择它们，然后按 Delete 键或右击选择 Delete 选项。

1.3.9 MATLAB 帮助

你有三种方法可以得到**MATLAB**的帮助。最好的方法是使用帮助空间窗口（helpbrowser）。你可以单击**MATLAB**桌面工具栏上的图标，也可以在命令窗口（TheCommand Windows）中输入 helpdesk 或 helpwin 来启动帮助空间窗口（helpbrowser）。你可以通

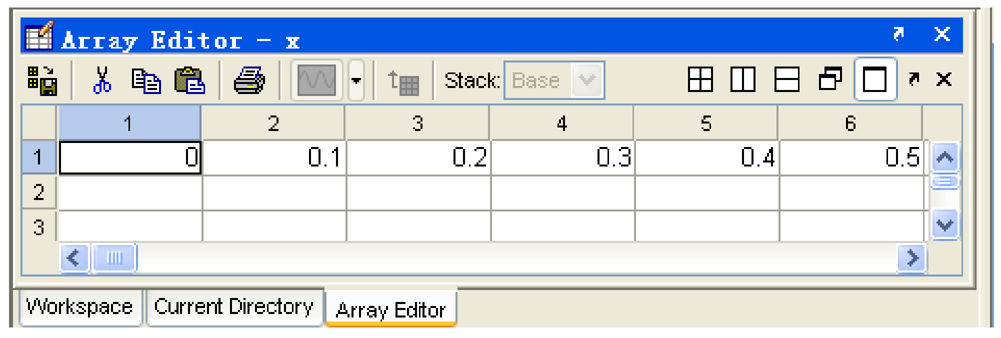


图 1.8 双击工作空间内的一个变量可调用数组编辑器（ArrayEditor）。数组编辑器（ArrayEditor）允许用户改变变量和数组的值。

过浏览 **MATLAB** 参考证书或搜索特殊命令的细节得到帮助。帮助空间窗口如图 1.9 所示。

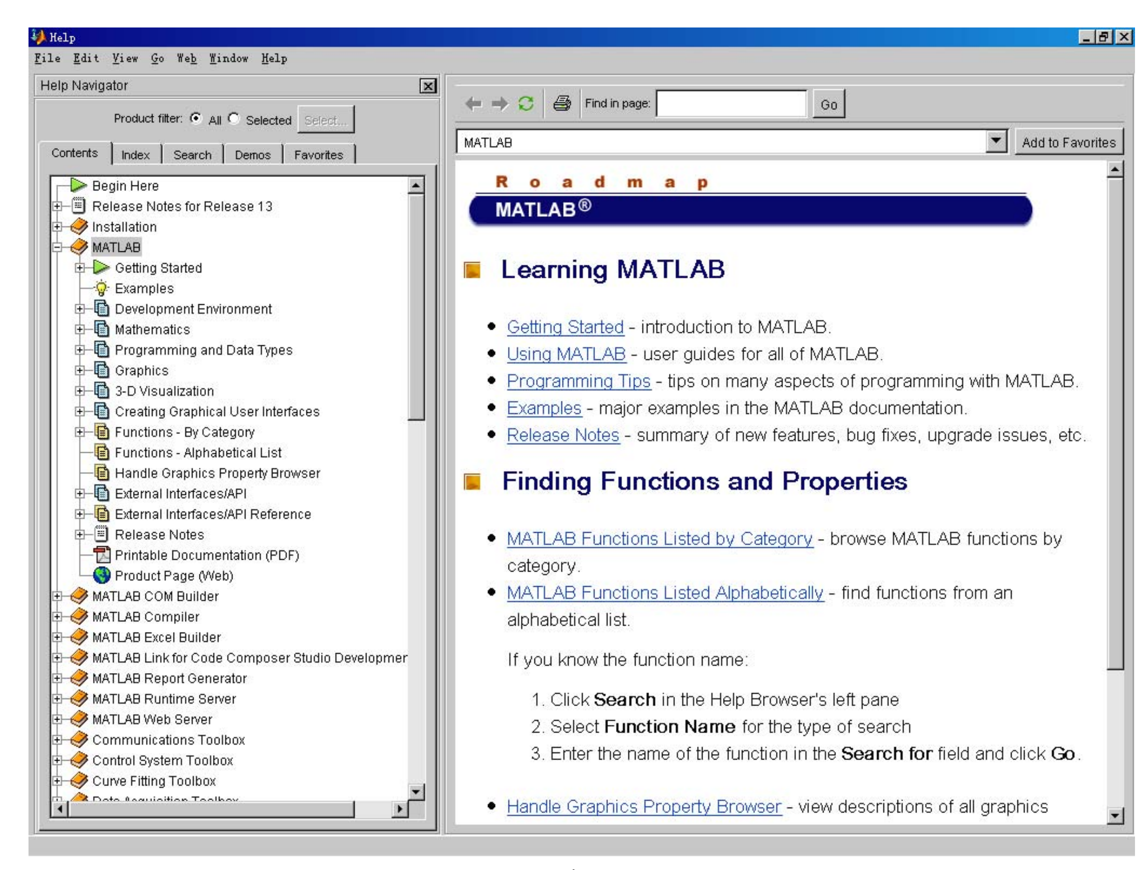


Figure1.9 帮助窗口

另外还有两种运用命令行的原始形式得到帮助。第一种方法是在 **MATLAB** 命令窗口(The Command Windows)中输入 help 或 help 和所需要的函数的名字。如果你在命令窗口(TheCommandWindows)中只输入 help，**MATLAB** 将会显示一连串的函数。如果有一个专门的函数名或工具箱的名字包含在内，那么 help 将会提供这个函数或工具箱。

第二种方法是通过 lookfor 命令得到帮助。lookfor 命令与 help 命令不同，help 命令要求与函数名精确匹配，而 lookfor 只要求与每个函数中的总结信息有匹配。Lookfor 命令

比 help 命令运行起来慢得多，但它提高了得到有用信息的机会。举个例子，假设你想找到一个求矩阵的逆阵（inverseofmatrix）的函数。但是 **MATLAB** 中没有叫 inverse 的函数，这时 help 命令就不起作用了，只能用 lookfor 命令，得到以下结果：

```
>> lookfor inverse
INVHILB Inverse Hilbert matrix.
IPERMUTE Inverse permute array dimensions.
ACOS      Inverse cosine.
ACOSD     Inverse cosine, result in degrees.
ACOSH     Inverse hyperbolic cosine.
ACOT      Inverse cotangent.
ACOTD     Inverse cotangent, result in degrees.
ACOTH     Inverse hyperbolic cotangent.
ACSC      Inverse cosecant.
ACSCD     Inverse cosecant, result in degrees.
ACSCH     Inverse hyperbolic cosecant.
ASEC      Inverse secant.
ASECD     Inverse secant, result in degrees.
ASECH     Inverse hyperbolic secant.
ASIN      Inverse sine.
ASIND     Inverse sine, result in degrees.
ASINH     Inverse hyperbolic sine.
ATAN      Inverse tangent.
ATAN2     Four quadrant inverse tangent.
ATAND     Inverse tangent, result in degrees.
ATANH     Inverse hyperbolic tangent.
ERFCINV Inverse complementary error function.
ERFINV Inverse error function.
INV       Matrix inverse.
PINV      Pseudoinverse.
IFFT Inverse discrete Fourier transform.
IFFT2 Two-dimensional inverse discrete Fourier transform.
IFFTN N-dimensional inverse discrete Fourier transform.
IFFTSHIFT Inverse FFT shift.
inverter.m: %% Inverses of Matrices
DRAMADAH Matrix of zeros and ones with large determinant or inverse.
INVHESS Inverse of an upper Hessenberg matrix.
```

通过这个列表我们可以看到我所需的函数的名字为 inv.

1.3.10 一些重要的命令

如果你是个 **MATLAB** 新手，一些示例可能有助你理解它的功能。在命令窗口(The Command Windows)中输入 demo 或在启动平台中选择“demos”来运行 **MATLAB** 内建的示例。在任何时侯你都可以用 clc 命令清空命令窗口(The Command Windows)中的内容，可以用 clf 清空当前图象窗口中的内容。在工作空间窗口（The workspace browser）中变量可用 clear 命令清除。正如我们看到的，工作空间窗口（The workspace browser）中的变量在独立的命令和 M 文件间执行时，可能会出现第一个问题中的变量存留在工作区而影响到第二个问题的解决。为了避免这种情况的发生，在新的计算开始之前，应当有 clear 命令清空工作区。

另一个重要的命令是 abort 命令。如果一个 M 文件运行时间过长，里面可能含有无限循环，而没有结束。在这种情况下，可在命令窗口内输入 control-c(简写为^c)。输入这个命令方法是光标在命令窗口内，按住控制键然后按 c。当 **MATLAB** 删除了^c,说明这个程序已经停止并回到命令行提示符状态。省略号（!）是另一个重要的特殊字符。它的特殊作用是给计算机操作系统发送一个命令。在省略号后的字符会发送给计算机并且执行，如果在计算机的命令行提示符中输入字符是一样的。这种特性使系统命令更容易植入**MATLAB**程序中。

最后，你能用 diary 命令记录下在 **MATLAB** 中运行过程中每个线程所做的事。命令的格式如下：

```
diary filename
```

当这个命令被执行后，所有在命令窗口(The Command Windows)中的输入和输出将会被记录在 diary 文件中。这是一个非常重要的工具，当 **MATLAB** 发生错误而中断时，利用它你可以重建重要的事件。diary off 命令中止写入 diary 文件，diary on 命令重新开始写入。

1.3.11 MATLAB 搜索路径

MATLAB 用 **MATLAB** 搜索条寻找 M 文件。在你的文件系统中，**MATLAB** 的 M 文件是以目录的形式被组织。

如果用户在 **MATLAB** 提示符后输入一个名字，那么 **MATLAB** 在解释器将按以下顺序寻找这个名字：

1. 它先查看这个名字是否是个变量名。如果它是一个变量，**MATLAB** 将会显示出这个变量的值。
2. 然后检查看它是否是内建函数或命令。如果是，则执行对应的函数或命令。
3. 检查是不是在当前目录下的一个 M 文件。如果是，则执行对应的函数或命令。
4. 检查是不是在 **MATLAB** 搜索路径的所有目录下的一个 M 文件。如果是，则执行对应的函数或命令。

注意：如果首先检测到的是变量名，且这个变量名与 **MATLAB** 的某一个函数或命令同名，那么这个函数或命令将变得无法访问。这是初学者易犯的错误之一。

编程隐患
如果变量名与 MATLAB 中的函数或命令重名，那么这个函数或命令将不能被访问。

还有，如果有多个函数或命令重名，那么 **MATLAB** 将会执行在搜索路径中找到的第一个，其他的将不会被执行。对于初学者，这也是一个常见的问题，往往将 M 文件的名字与 **MATLAB** 内建函数或命令重名，从而导致函数或命令的不能访问。

编程隐患
不要创建和 MATLAB 内建函数或命令同名的 M 文件。

MATLAB 还包括一个特殊的命令——which 命令，它能帮助我们找到正在执行的文件版本和它的路径。在检查文件名冲突方面它是非常有用的。这个命令的格式是

```
which filename
```

filename 代表你所要加载的函数名。举个例子，你要加载的函数是 cross.m:

```
>> which cross
D:\MATLAB7\toolbox\MATLAB\specfun\cross.m
```

我们可以运用启动平台中的路径工具（the path tool）随时检查和修改这个路径，或者在命令窗口(The Command Windows)中输入 editpath 命令。路径工具（the path tool）

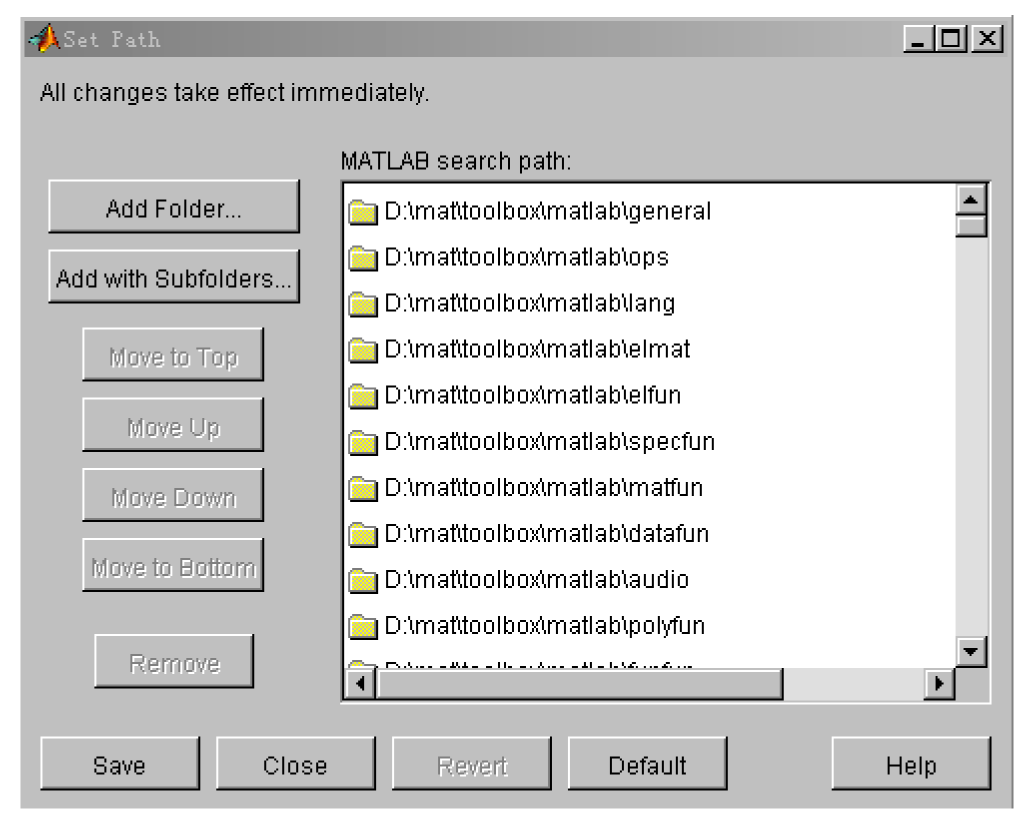


图 1.10 路径工具（the path tool）

如图 1.10 所示。它允许使用者添加，删除路径和改变在目录中的顺序。还有其他与路径相关的函数，包括：

- addpath　　增加目录到 **MATLAB** 搜索路径。
- path　　显示 **MATLAB** 搜索路径。
- path2rc　　增加当前目录到 **MATLAB** 搜索路径。
- rmpath　　移动 **MATLAB** 搜索路径中的目录。

1.4 把 MATLAB 当作便笺薄来使用

MATLAB 可用作便笺薄以最简单的形式进行数学计算。所要进行的计算可直接输入命令窗口(The Command Windows)，+，-，*，/和^分别代表加号，减号，乘号，除号和乘方。在窗口中输入一个表达式后，它将会自动计算和显示出结果。例如，你要计算一个半径为 r，高度为 1 的圆柱体的体积。圆柱体的底面面积的计算式为

$$A=\pi r^2 \qquad (1.1)$$

圆柱体的体积计算式为

$$V=Al \qquad (1.2)$$

假设一个圆柱体的底面半径为 0.1m，高度为 0.5m，这个圆柱体的体积可以通过以下 **MATLAB** 语言进行计算得到：

```
>> A=pi*0.1^2
A =
    0.0314
>> V=A*0.5
```

```
V =
    0.0157
```

注意 pi 的预定义值为 3.1415926…，还有 A 的值存在于 **MATLAB** 中，当计算 V 时 A 值得到重复利用。

测试 1.1

本测试提供了一个快速的检查方式，看你是否掌握了第一章的基本内容。如果
你对本测试有疑问，你可以重读本章，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. **MATLAB** 的命令窗口的作用是什么？编辑/调试窗口？图象窗口？
2. 列出几种不同的得到 **MATLAB** 帮助的方法。
3. 什么是工作区？在同一工作区内，你如何决定它里面存储了什么？
4. 你怎样清空 **MATLAB** 工作区内的内容？
5. 一小球从空中下落的位移公式为

$$x = x_0 + v_0 t + \frac{1}{2}at^2$$

用 **MATLAB** 计算小球在 t=5s 时的位置，已知 x_0 =10m，v_0 =15m/s， a=-9.84m/s^2.

6. 假设 x=3，y=4。用 **MATLAB** 计算下列表达式：

$$\frac{x^2 y^3}{(x-y)^2}$$

下列问题将有助于你熟悉 **MATLAB** 工具。(如果你使用的 **MATLAB** 比 6.0 旧，那么解决这些问题将比较麻烦，因为旧版本的 **MATLAB** 的许多工具与新版本的不同)

7. 在命令窗口(The Command Windows)执行 M 文件 calc_area.m 和 sin_x.m(这些 M 文件可在本书的网站上得到)。然后用工作空间平台查看有那些变量在当前工作区。
8. 用数组编辑器查年和修改变量 x 的值。然后的在命令窗口键入命令 plot(x,y)中，观察在图象窗口内数据怎样被显示。

1.5 总结

在本章中，我们学到了基本类型的 **MATLAB** 窗口，工作区和如何得到在线帮助。当 **MATLAB** 程序启动时, **MATLAB** 桌面就会被显示出来。在单一位置它集成许多的 **MATLAB** 工具。这些工具包括命令窗口(The Command Windows)，命令历史窗口，启动平台，数组编辑器，和当前目录查看器。命令窗口是最重要的窗口，因为所有的命令都得在此键入，所有的结果在此输出。

用编辑/调试器经常用于创建和修改 M 文件。它用于显示 M 文件的内容，内容用不同的颜色显示出来：解释，关键字，字符串等等。

图象窗口用于显示图象。

MATLAB 用户可以通过帮助空间窗口，help 命令，lookfor 命令三种方式得到帮助。帮助空间窗口可以帮助用户访问所有的 **MATLAB** 文件设置。而 help 命令是一个在命令窗口内显示帮助的方法。不幸的是，运用这个命令你必须知道你所要帮助的函数的名字。只要 **MATLAB** 函数评论的第一评论句中的单词与搜索词相对应，那么用 lookfor 命令就可找到，并显示出来。

当用户在命令窗口中键入一个命令，**MATLAB** 按照 **MATLAB** 路径特殊编排而成的目录搜索这个命令。当 **MATLAB** 执行了路径中第一个与之对应的 M 文件，其他的拥有相同名字的 M 文件将会被忽略。路径工具（the path tool）允许使用者添加，删除路径和改变在目录中的顺序。

1.5.1 MATLAB 总结

下面的总结表是本章的遇到的所有的 **MATLAB** 的特殊符号，并带有简短的解释。

特殊符号

+	加号
-	减号
*	乘号
/	除号
ˆ	乘方

1.6 练习

1.1 下列 **MATLAB** 语句用于画出函数 $y(x) = 2e^{-0.2x}$ 在[0,10]的值。

```
x=0:0.1:10;
y=2*exp(-0.2*x);
plot(x,y);
```

用 **MATLAB** 编辑器创建一个新的 M 文件，把上面的语句写入这个文件并命名为 test1.m。然后在命令窗口中输入 test1 执行这个文件。看得到什么结果？

1.2 通过以下两种方式得到关于 exp 函数的帮助

（a）在命令窗口中输入 help exp 命令

（b）运用帮助空间窗口

1.3 使用 lookfor 命令寻找一个数以十为底的对数函数。

1.4 假设 u=1 和 v=3，用 **MATLAB** 语句编写下列语句

a. $\frac{4u}{3v}$ b. $\frac{2v^{-2}}{(u+v)^2}$ c. $\frac{v^3}{v^3-u^3}$ d. $\frac{4}{3}\pi v^2$

1.5 应用 **MATLAB** 帮助空间窗口查找显示文件当前目录的命令。**MATLAB** 启动时它文件当前目录是什么？

1.6 应用 **MATLAB** 帮助空间窗口创建一个新目录，这个在当前目录下这个新目录的名字为 mynewdir.把这个目录置于路径的顶端。

1.7 把当前目录改为 mynewdir 目录，然后打开一个编辑窗口，增加以下语句：

```
% create an input array from -2*pi to 2*pi
t = -2*pi:pi/10:2*pi;
% calculate|sin(t)|
x=abs(sin(t));
%plot result
plot(t,x);
```

把此文件以 test2.m 为文件名保存，然后在命令窗口中输入 test2 执行此文件。

1.8 关闭画图窗口，然后返回到原始目录，然后再命令窗口中输入 test2。看有何情况发生，为什么。

第二章 MATLAB 基础

在本章我将向大家介绍 **MATLAB** 的基本元素。在本章的章末，你将会编写简单的函数化的工具。

2.1 变量和数组

MATLAB 程序的基本数据单元是数组。一个数组是以行和列组织起来的数据集合，并且拥有一个数组名。数组中的单个数据是可以被访问的，访问的方法是数组名后带一个括号，括号内是这个数据所对应行标和列标。标量在 **MATLAB** 中也被当作数组来处理——它被看作只有一行一列的数组。

数组可以定义为向量或矩阵。向量一般来描述一维数组，而矩阵往往来描述二维或多维数组。在本书中，当我们讨论一维数组时用向量表示，当我们讨论二维或多维向量时用矩阵。如果在特殊情况下，同时遇到这两种数组，我们就把他们通称为“数组”。

数组的大小（size）由数组的行数和列数共同决定,注意行数在前。一个数组所包含的数据多少可由行数乘列数得到。例如，下列数组的大小为

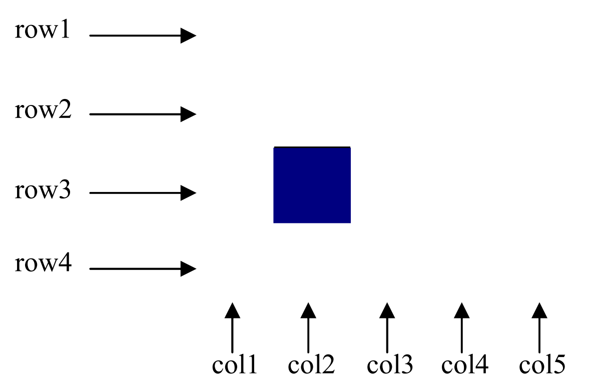


图 2.1 一个数组是以行和列组织起来的数据集合,此数组 arr 含有 20 个元素，共 4 行，5 列。阴影元素是 arr(3,2)

数组	大小
$A=\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix}$	这是一个 3×2 矩阵，包含 6 个元素
$B=\begin{bmatrix}1&2&3&4\end{bmatrix}$	这是一个一维行向量，共有 4 个元素

$C=\begin{bmatrix}1\\2\\3\\4\end{bmatrix}$	这是一个一维行向量，共有 4 个元素

数组中的单个数据是可以被访问的，访问的方法是数组名后带一个括号，括号内是这个数据所对应的行标和列标。如果这个数组是一个行向量或列向量，则只需要一个下标。例如上面的数组 A（2 1）为 3，C（2）为 2。一个 **MATLAB** 变量是一段包含一个数组的内存区，并且拥有一个用户指定的变量名。通过适当的命令和它的变量名随时可以就调用它和修改它。

MATLAB 的变量名必须以字母开头，后面可以跟字母，数字和下划线（_）.只有前 31 个字符是有效的；如果超过了 31 个字符，基余的字符将被忽略。如果声明两个变量，两变量名只有第 32 个字符不同，那么 **MATLAB** 将它们当作同一变量对待。

编程隐患

确保你所声明的变量名前 31 个字符是独一无二的。否则，MATLAB 将无法辨认出它们的不同。

当你编写程序时，给变量起一个有意义的名字非常的重要。有意义的名字极大的提高了程序的可读性和可维护性。像 day，month 和 year 这样的名字意义非常明确，即使第一次看到也能理解。尽管空格不能用在 **MATLAB** 变量名中，但是可以用下划线代替空格创造出有意义的变量名。比如，changerate 可以写成 change_rate.

好的编程习惯

给你的变量起一个描述性的且易于记忆的变量名。例如，货币汇率可以 exchange_rate 为变量名。这种方法将使得你的程序更加明确且易于理解。

在你所写的程序的开头列出一数据字典（data dictionary）十分的重要.数据字典列举了你在本程序中用到的所有变量的定义。它的定义应包括本条目的所要描述的内容和它在执行时所在的单元。当编写程序时，编定数据字典看似没有必要。但是设想一下，在过了一段时间后，你或其他人要对此程序修改，这时数据字典就显得十分的有用。

好的编程习惯

给每个程序创建一个数据字典以增强程序的可维护性。

在 **MATLAB** 语言中是区分字母大小的，也就是说，大写字母和小写字母代表的东西是不同的。所以变量 NAME,Name，name 在 **MATLAB** 中是不同的。所以已用过的小写变量名与一个新建大写的变量名重名，这时使用时要特别地小心。在一般情况下，我们一律用小写字母来表示。

好的编程习惯

在每次用到一个变量时，我们要确保变量名的大小写的精确匹配。在变量名中只使用小写字母是一个好的编程习惯。

两个最常见的变量类型是 char 型和 double 型。double 型的变量包括由 64 位双精度浮点数构成的标量或数组。这种变量可以代表实数，虚数和复数。每个值的实部和虚部的变化范围为正负 10^{-308}~10^{308}，拥有 15 到 16 位有效数字。这是基本的数字数据类型。

无论什么时侯，你将一个数值赋值于一个变量名，那么 **MATLAB** 将自动建立一个 double 型变量。例如，下面语句创建了一个以 var 为变量名的 double 型变量，包含了一个 double 型的单个元素，存储了复数值（1+i）；

```
var=1+i;
```

char 型的变量包括由 16 位数值构成的标量或数组，每一个 16 位数代表一个字符。这个类型的经常用于字符串操作，当一个字符或字符串赋值于一个变量名时，系统会自动建立一个 char 型变量。例如，下面的这个语句创建了一个 char 型变量 comment，并存储了一个字符串在其内。当这个语执行后，系统将会建立一个 1×26 的字符串数组。

```
comment='this is a character string';
```

像 C 语言这样的语言中，变量类型和变量在使用之前必须强制声明。这种语言我们叫它**强类型语言**。相对地，像 **MATLAB** 这样的叫做**弱类型语言**。通过简单的赋值形式就可以创建变量，变量类型取决于创建时的类型。

2.2 MATLAB 变量的初始化

当变量初始化时，**MATLAB** 将会自动建立变量。有三种方式初始化 **MATLAB** 中的变量：

1．用赋值语句初始化变量

2．用 input 函数从键盘输入初始化变量

3．从文件读取一个数据

前两种方法我们在这里讨论，第三方法我们将在 2.7 节介绍。

2.2.1 用赋值语句初始化变量

最简单的创建和初始化一个变量的方法是用赋值语句赋予变量一个或多个值。赋值语句的一般形式如下

```
var = expression
```

var 是变量名，expression 可以是一个标量、一个数组或常量、其他变量和数学运算符号（+、-）的联合。这个表达式（expression）的值是通过一般的数学运算法则计算出来的，然后将产生的结果存储到变量 var 中。下面是一些用赋值语句初始化的变量：

```
var=40*i;
var2=var/5;
array=[1 2 3 4];
x=1;
y=2;
```

第一个例子创建了一个 double 类型的标量变量，存储了一个虚数 40i。第二个例子创建了一个表达式 var2，把 var/5 的值存储于内。第三个例子创建了一个数组 array，并存储了一个 4 元素的行向量于内。最后一个例子显示了多个赋值语句可写在同一行，中间用逗号或分号隔开。注意如果在赋值语句执行时变量已经存在，那么这个变量原有的值将被覆盖。

正如第三个例子显示的，数据数组也可以初始化变量。我们可以用是括号（）和分号建立数组。所有元素按行阶排序，换句话说，每一行的值从左向右，顶部的行置于最前，底部的行置于最后。在一行内单个数值可用空格或逗号隔开，而行与行之间要与则用分号隔开，或另起一行书写。下面的表达式都是合法的，能用于建立一个变量：

[3.4]	这个表达式创建了 1×1 数组(一个标量),包含数值 3.4.这时括号可以省略.
[1.0 2.0 3.0]	这个表达式创建了 1×3 数组,一维行向量[1 2 3]

[1.0;2.0;3.0]	表达式创建了一个 3×1 数组,一维列向量$\begin{bmatrix}1\\2\\3\end{bmatrix}$
[1,2,3;4,5,6]	这个表达式创建了一个 2×3 数组,矩阵$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$
[1,2,3 4,5,6]	这个表达式创建了一个 2×3 数组,矩阵$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$
[]	是个空数组,没有行,没有列(注意他与元素全为零的数组的区别)

注意一个数组每一行元素的个数必须完全相同,每一列元素的个数也必须完全相同.像

```
[1 2 3;4 5];
```

这样的表达式是非法的,因为第一行有 3 个元素,第二行有只有 2 个元素.

编程隐患

每一行元素的个数必须完全相同,每一列元素的个数也必须完全相同.试图创建一个不同行(列)拥有不同数目元素的数组,在编译时将会出现错误.

用于初始化数组的表达式可以包括代数符号或事先已经定义好的数组.例如赋值语句

```
a=[0 1+7]
b=[a(2) 7 a]
```

定义了数组 a=[0 8]和数组 b=[8 7 0 8].

当我们创建一个数组时,不是每一个元素都必须定义.如果要定义一个特殊的数组,或只有一个或几个元素没有定义,那么之前的那些元素将会自动创建,并初始化为 0.例如,如果数组 c 事先没有定义,语句 c(2,3)=5 将会创建一矩阵 $c=\begin{bmatrix}0&0&0\\0&0&5\end{bmatrix}$。相似地，指定一个值赋予一个存在的数组，但超过了这个数组的大小。例如，假设存在一数组 d=[1 2],下面这个语句

```
d(4)=4;
```

将会制造出数组 d=[1 2 0 4].

在每个赋值语句末的分号有特殊的目的：无论在何时一个表达式在赋值语句中被赋值，分号将会中止变量值的重复。如果句末没有分号，变量值将会自动显示在命令窗口(The Command Windows)中。

```
>> e=[1 2 3;4 5 6]
e =
     1     2     3
     4     5     6
```

如果在赋值语句末有分号，这种重复将会消失。重复是一个用于检查你的工作极好的方法，但是它降低了运行速度。因此，我们在一般情况下总是禁止重复。尽管如此，重复计算的结果提供了一个强大的应急调试器。如果你不能确定一个特定的赋值语句结果是多少，这时你可以去掉这个语句后的分号，当这个语句编译时，结果会显示在命令窗口(The Command Windows)。

好的编程习惯

在 MATLAB 赋值语句后加上一个分号来禁止变量值在命令窗口(The Command Windows)的重复。这将大大提高编译的速度。

好的编程习惯

如果你在调试程序时需要检测一个语句的结果，可能把句后的分号去掉，这样结果将会出现在命令窗口(The Command Windows)。

2.2.2 用捷径表达式赋值

创建一个小数组用一一列举出元素的方法是比较容易的，但是当我们创建包括成千上万个元素的数组怎么办？把每一个元素列举出来则不太现实。

MATLAB 提供一种专门的捷径标记法，这种方法用克隆运算符（colon operator）适用于上述情况。克隆运算符指定一系列的数值，它指定了这个系列数的第一值，步增和最后一个值。它的一般顺序始下

```
first:incr:last
```

first 代表数组的每一个值，incr 代表步增量，last 代表这个数组的最后一个值。如果步增量为 1，那么步增量可省略，而变成了 first:last 格式。

例如，表达式 1:2:10 是创建一个 1×5 行向量[1 3 5 7 9]的简便方法。

```
>> x=1:2:10
x =
     1     3     5     7     9
```

用克隆标记法初始化一个含有一百个元素的数组 $\left[\frac{\pi}{100},\frac{2\pi}{100},\cdots,\pi\right]$，语句如下

```
Angles = (.01:.01:1)*pi
```

捷径表达式可以联合转置运算符（’）来初始化行向量，或更加复杂的矩阵。转置运算符可以在需要的情况下完成行和列的转换。因为这个表达式

```
f = [1:4]';
```

产生一个 4 元素行向量[1 2 3 4]，然后将其转换成 4 元素的列向量 $f=\begin{bmatrix}1\\2\\3\\4\end{bmatrix}$;相似地，表达式

```
g = 1:4;
h = [g' g']
```

将会创建一个矩阵 $h=\begin{bmatrix}1&1\\2&2\\3&3\\4&4\end{bmatrix}$

2.2.3 用内置函数来初始化

数组也可以用 **MATLAB** 内置函数创始化。例如，函数 zeros 可以初始化任何大小的全为零的数组。用许多形式的 zeros 函数。如果这个函数的参数只是一个标量，那么 **MATLAB** 将会创建一个方阵，行数和列数均为这个参数。如果这个函数有两个标量参数，那么第一个参数代表行数，第二个参数代表列数。因为 size 函数所返回的一个数组的行数和列数，所以它可以联合 zeros 函数来创建一个相同大小零矩阵。下面是一些用到 zeros 函数的例子。

```
a = zeros(2);
b = zeros(2,3);
c = [1 2;3 4];
d = zeros(size(c))
```

这些语句产生下列的数组

$$a = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$c = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

相似地,ones 函数产生的数组包含的元素全为 1,eye 函数通常用来产生单位矩阵,只有对角线的元素为 1.其他元素为 0.表 2.1 列出一些用于初始化变量的函数.

表 2.1 用于创始化变量的 **MATLAB** 函数

函数	作用
zeros(n)	创建一个 n×n 零矩阵
zeros(n,m)	创建一个 n×m 零矩阵
zeros(size(arr))	创建一个与数组 arr 的零矩阵
ones(n)	创建一个 n×n 元素全为 1 矩阵
ones(n,m)	创建一个 n×m 元素全为 1 矩阵
eye(n)	创建一个 n×n 的单位矩阵
eye(n,m)	创建一个 n×m 的单位矩阵
length(arr)	返回一个向量的长度或二维数组中最长的那一维的长度
size(arr)	返回指定数组的行数和列数

2.2.4 用关键字 input 初始化变量

关键字 input 用来提示使用者和直接从键盘输入初始化变量.当脚本文件(Script files)时,它可以用来提示使用者输入.input 函数在命令窗口(The Command Windows)显示提示语句,并等待用户输入一个值.例如,下面的赋值语句:

```
my_val = input('Enter an input value:')
```

当这个语句被编译时,**MATLAB** 打印出字符串 enter an input value:,然后等待用户回复.如果要只输入一个数,那么只需要直接键入,如果要输入一个数组,则必须带上中括号([]).不管怎样，当按下回车键时.在窗口输入的任何值都会被储入变量 my_val.如果只按下回车键,那么这个变量就存储了一个空矩阵.

如果 input 函数中有字符's'做为它的第二个参数,输入的数据就被当字符串.因此,语句

```
>> in1 = input('enter data:');
Enter data:1.23
```

把数值 1.23 存储到 in1 中.而语句

```
>> in2 = input('enter data:','s')
Enter data:123
```

把字符串 1.23 存储到 in2 中.

测试 2.1

本测试提供了一个快速的检查方式，看你是否掌握了 2.1 和 2.2 的基本内容。如果你对本测试有疑问，你可以重读 2.1 和 2.2，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. 数组,矩阵,向量有什么区别?
2. 回答关于下列矩阵的有关问题

$$C=\begin{bmatrix}1.1 & -3.2 & 3.4 & 0.6\\ 0.6 & 1.1 & -0.6 & 3.1\\ 1.3 & 0.6 & 5.5 & 0.0\end{bmatrix}$$

(a)C 的大小是多少？
(b)C(2,3)的值是多少？
(c)列出值为 0.6 的元素的下标

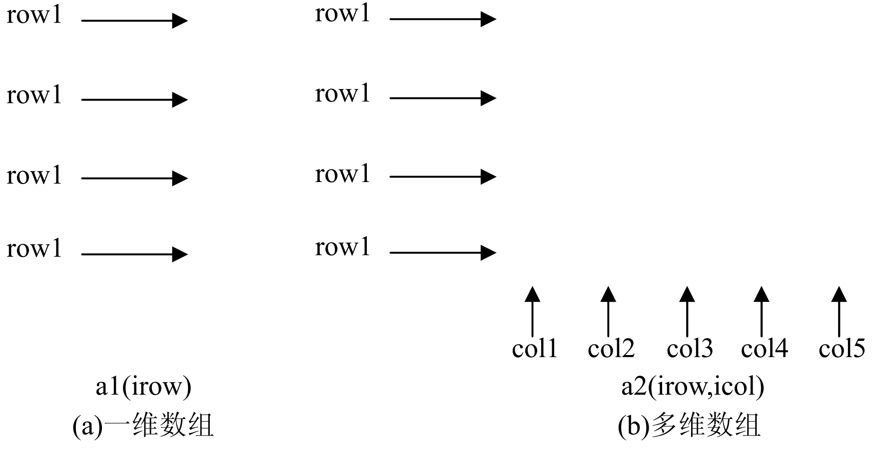
3. 确定下列数组的大小，通过 whos 或工作空间窗口（The workspace browser）检查你的答案。注意在本练习中后面的数组可能要用到前面数组的定义。

(a) u=[10 20*i 10+20]
(b) v=[-1;20;3]
(c) w=[1 0 -9;2 -2 0;1 2 3]
(d) x=[u' v]
(e) y(3,3)=-7
(f) z=[zeros(4,1) ones(4,1) zeros(1,4)']
(g) v(4)=x(2,1)

4. w(2,1)的值是多少？
5. x(2,1)的值是多少？
6. y(2,1)的值是多少?
7. 当语句（g）执行后，v(3)的值是多少？

2.3 多维数组

正如我们所看到的，**MATLAB** 的数组可能是一维或多维的。一维的数组可以形象地看作一系列的数垂直地罗列起来，用一个下标就可以调用数组中的元素（如图 a）。这样的数组适用于一个变量的函数，例如在规定的时间间隔后一系列的测量温度。



(a)一维数组　　(b)多维数组

图 2.2

许多数据的类型需要多变量的函数。例如，我要在 5 个不同的地方，每个地方测 4 次温度。在这种情况下，我们的 20 次测量结果在逻辑上分为五个不同的行，每行有 4 个测量结果（如图 b）.在这种情况下，我们就需要两个下标来调用这个数组特定的函数：第一个下标选择行，第二个下标选择列。这样的数组叫做**二维数组**。二维数组中元素的个数取决于这个数组的行数和列数。

出于问题的需要，**MATLAB** 允许我们创建多维数组。这些数组的每一维对应一个下标，和每一个单个元素都可以通过它的每一个下标被调用。在这个数组中元素的总和取决于每一维中元素的个数。例如，下面两个语句创建了一个 2×3×2 数组 c

```
>> c(:,:,1)=[1,2,3;4,5,6];
```

```
>> c(:,:,2)=[7,8,9;10,11,12];
>> whos c
  Name        Size                    Bytes  Class

  c           2x3x2                      96  double array
```

这个数组（2×3×2）包括 12 种元素，它的内容显示方法和其他数组的显示方法大体相同

```
>> c
c(:,:,1) =
     1     2     3
     4     5     6
c(:,:,2) =
     7     8     9
    10    11    12
```

2.3.1 多维数组在内存中的存储

一个有 m 行和 n 列的二维数组包括 m×n 个元素，这些元素在计算机的内存中将会占有 m×n 个连续的内存空间。这些数组的元素在内存中是如何排列的呢？**MATLAB 以列主导顺序**分配数组中的元素。也就是说，内存先分配第一列的元素，然后第二列，第三列，……以此类推，直到所有列都被分配完。图 2.3 说明 4×3 数组 a 的内存分配情况。正如我们所看到的，元素 a(1,2)是其实在内存分配的第五个位置。在下一节我们讨论用单一下标访问数组元素和第八章低级 I/O 接口，内存分配元素的顺序将变得十分重要。

1	2	3
4	5	6
7	8	9
10	11	12

(a)

...	
1	a(1,1)
4	a(2,1)
7	a(3,1)
10	a(4,1)
2	a(1,2)
5	a(2,2)
8	a(3,2)
11	a(4,2)
3	a(1,3)
6	a(2,3)
9	a(3,3)
12	a(4,3)
...	

(b)

图 2.3(a)数组 a 中的数据 (b)数据 a 在内在中的布局

这种分配方式也适用于多维数组。数组的第一个下标增长最快，第二个仅次之，依此类推，最后一个变化最慢。例如，在一个 2×2×2 数组中，它的元素在内存中的分配顺序是

（1，1，1），（2，1，1），（1，2，1），（2，2，1），（1，1，2），（2，1，2），（1，2，2），（2，2，2）。

2.3.1 用单个下标访问多标数组

MATLAB 的特性之一就是它允许使用者或程序员把一个多维数看作一个一维数组，这

个一维数组的长度等于多维数组的元素数。如果用一个下标访问一个多维数组，那么元素的排列顺序就是内存的分配顺序。

例如，假设我们要声明一个 4×3 的数组如下：

```
>> a = [1 2 3; 4 5 6; 7 8 9; 10 11 12]

a =

     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

那么 a(5)的值为 5 和 a(1,2)的值相同，这是因为元素 a(1,2)排在内存第五个位置。

在一般情况下，我们不应使用 **MATLAB** 的这一特性。用单个下标访问多维数组可能会带很多的麻烦。

好的编程习惯
在访问多维数组时，总是使用合适的维数。

2.4 子数组

你可以选择和使用一个 **MATLAB** 函数的子集，好像他们是独立的数组一样。在数组名后面加括号，括号里面是所有要选择的元素的下标，这样就能选择这个函数的子集了。例如，假设定义了一个数组 arr1 如下

```
arr1 = [1.1 -2.2 3.3 -4.4 5.5]
```

那么 arr1(3)为 3.3,arr1([1 4])为数组[1.1 -4.4],arr1(1:2:5)为数组[1.1 3.3 5.5].

对于一个二维数组，克隆运算符可以用于下标来选择子数组。例如，假设

```
arr2 = [1 2 3; -2 -3 -4;3 4 5]
```

将建立一个数组

$$arr2=\begin{bmatrix}1 & 2 & 3\\ -2 & -3 & -4\\ 3 & 4 & 5\end{bmatrix}$$

在这种定义下，子数组 arr2(1,:)为[1 2 3]，子数组 arr2(:,1:2:3)为$\begin{bmatrix}1 & 3\\ -2 & -4\\ 3 & 5\end{bmatrix}$

2.4.1 end 函数

MATLAB 中有一个特殊的函数叫做 end 函数,对于创建子数组的下标非常的有用.当用到一个函数的下标时,end 函数将会返回下标的最大值.

例如,假设数组 arr3 定义如下:

```
arr3 = [1 2 3 4 5 6 7 8];
```

那么 arr3(5:end)将会产生数组[5 6 7 8],arr3(end)将会产生值 8.

end 函数返回的值一般为所要下标的最大值.如果 end 函数显示有不同的下标,那它将在一个表达式内返回不同的值.例如,假设一个 3×4 数组 arr4 的定义如下:

```
arr4 = [1 2 3 4;5 6 7 8;9 10 11 12]
```

那么表达式 arr4(2:end,2:end)将会返回$\begin{bmatrix} 6 & 7 & 8 \\ 10 & 11 & 12 \end{bmatrix}$.注意第一个 end 返回值为 3,第二个返回值为 4.

2.4.2 子数组在左边的赋值语句的使用

只要数组的形（行数和列数）和子数组的形相匹配，把子数组放于赋值语句的左边用来更新数组中的值。如果形不匹配，那么将会有错误产生。例如，下面有一个 3×4 数组定义如下：

```
>> arr4 = [1 2 3 4;5 6 7 8;9 10 11 12]

arr4 =

     1     2     3     4
     5     6     7     8
     9    10    11    12
```

因为在等号左边的表达式的形（2×2）与 a 相匹配，那么下面的这个赋值语句是合法的。

```
>> arr4(1:2,[1 4])=[20 21;22 23]

arr4 =

    20     2     3    21
    22     6     7    23
     9    10    11    12
```

注意数组元素（1，1），（1，4）（2，1）和（2，4）得到了更新。相对而言，两边的形不相匹配，则表达式是非法的，例如下面这个表达式。

```
>> arr5(1:2,[1 4])=[20 21]
??? Subscripted assignment dimension mismatch.
```

编程隐患
对于涉及子数组的赋值语句，等号两边的形必须相匹配。否则将会产生错误。

在 **MATLAB** 中用子数组赋值和用值直接赋值有很大的不同。如果用子数组赋值，那么只有相应的值得到更新，而其他的值保持不变。另一方面，直接赋值，则数组的原有内容全部删除并被新的值替代。例如，假设用一个数组 arr4 定义如下：

```
>> arr4 = [1 2 3 4;5 6 7 8;9 10 11 12]

arr4 =

     1     2     3     4
     5     6     7     8
     9    10    11    12
```

下面的赋值语句，只更新特定的元素：

```
>> arr4(1:2,[1 4]) = [20 21;22 23]

arr4 =

    20     2     3    21
    22     6     7    23
     9    10    11    12
```

相对地，下面的赋值语句更新了数组的全部内容，并改变了数组的形

```
>> arr4 = [20 21;22 23]

arr4 =

     20     21
     22     23
```

好的编程习惯
确保将赋值于子数组和赋值于数组。MATLAB 将它们当作两个不同的情况来对待。

2.4.3 用一标量来给子数组赋值

位于赋值语句的右边的标量值总是能匹配左边数组的形。这个标量值将会被复制到左边语句中所对应的元素。例如，假设用一个数组 arr4 定义如下：

```
arr4 = [1 2 3 4;5 6 7 8;9 10 11 12]
```

下面的表达式将一个值赋值于数组的 4 个元素。

```
>> arr4(1:2,1:2)=1
arr4 =
      1     1     3     4
      1     1     7     8
      9    10    11    12
```

2.5 特殊变量

在 **MATLAB** 中有许多预先定义好的特殊变量。在 **MATLAB** 中这些特殊变量可以随时使用，不用初始化。一些常见的预定义值列在表 2.2。

表 2.2 预定义特殊变量

函数	用途
pi	有 15 个有效值的 π
i,j	代表虚数 i($\sqrt{-1}$)
Inf	这个符号代表无穷大，它一般情况下是除以 0 产生的
NaN	这个符号代表没有这个数。它一般由数学运算得到的。例如，0 除以 0。
clock	这个特殊变量包含了当前的年，月，日，时，分，秒，是一个 6 元素行向量
date	当前的日期，使用的的字符形式，如 30-Dec-2007
eps	变量名是 epsilon 的简写。它代表计算能机辨别的两数之间的最小数
ans	常用于存储表达式的结果，如果一个结果没有明确的赋值给某个变量

这个些预定义值存储在一般的变量中，所以他们能被覆盖或改写。如果一个新值赋值于其中一个预定义变量，，那么这以后的计算中新值将会替代默认值。例如，考虑下面用于计算以半径为 10 的圆的周长的语句；

```
circl=2*pi*10
pi=3
circ2=2*pi*10
```

在第一个语句中，pi 有默认值 3.14159...，所以周长 6.28319 是正确的结果，第二个语句重定义 pi 为 3，所以第三个语句 circ2 为 60。在程序中修改预定义值会造成一些不正确的结果,并导致一些微小而难以发现的错误。设想一下，要在 1000 行的程序中找出一个像这样的

错误是多么不容易。

编程隐患
不要重定义有意义的预定义变量。否则将后患无穷，制造成出小而难以发现的错误。

测试 2.2

本测试提供了一个快速的检查方式，看你是否掌握了 2.3 到 2.5 的基本内容。如果你对本测试有疑问，你可以重读 2.3 和 2.5，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. c 数组的定义如下，写出下面子数组的内容。

```
c =
      1.1000    -3.2000     3.4000     0.6000
      0.6000     1.1000    -0.6000     3.1000
      1.3000     0.6000     5.5000          0
```

(a) c(2,:)　(b) c(:,end)　(c) c(1:2,2:end)　(d) c(6)
(e) c(4:end)　(f) c(1:2,2:4)　(g) c([1 4],2)　(h) c([2 2],[3 3])

2. 当赋值语句执行后，下列数组的内容是多少？

(a)
```
a=[1 2 3; 4 5 6; 7 8 9];
a([3 1],:)=a([1 3],:);
```
(b)
```
a=[1 2 3; 4 5 6; 7 8 9];
a([1 3],:)=a([2 2],:);
```
(c)
```
a=[1 2 3; 4 5 6; 7 8 9];
a=a([2 2],:);
```

3. 当数组执行后，下列数组 a 的内容是多少？

(a)
```
a=eye(3,3);
b=[1 2 3];
a(2,:)=b;
```
(b)
```
a=eye(3,3);
b=[4 5 6 ];
a(:,3)=b';
```
(c)
```
a=eye(3,3);
b=[7 8 9];
a(3,:)=b([3 1 2]);
```
(d)
```
a=eye(3,3);
b=[7 8 9];
a(3,:)=b([3 1 2]);
```

2.6 显示输出数据

在 **MATLAB** 中有许多的方法显示输出数据。最简单的方法是我们已经用过的去掉语句末的分号，它将显示在命令窗口(The Command Windows)中。在这里向大家介绍一些其他的方法。

2.6.1 改变默认格式

当数据重复在命令窗口(The Command Windows)时，整数以整形形式显示，其他值将以默认格式显示。**MATLAB** 的默认格式是精确到小数点后四位。如果一个数太大或太小，那么将会以科学记数法的形式显示。比如，语句

```
x = 100.11
y = 1001.1
z = 0.00010011
```

它的输入格式为

```
x =
   100.1100

y =
   1.0011e+003

z =
   1.0011e-004
```

改变默认输出格式要用到 format 命令，可根据表 2.3 改变数据的输出格式

表 2.3 输出显示格式

format 命令	结果	例子
format short	保留小数点后 4 位(默认格式）	12.3457
format long	保留小数点后 14 位	12.345678901234567
format short e	带有 5 位有效数字科学记数法	1.2346e+00
format short g	总共有 5 个数字，可以用科学记数法，也可不用	12.346
format long e	带有 15 位有效数字科学记数法	1.234567890123457e+001
format long g	总共有 5 个数字，可以用科学记数法，也可不用	12.3456789012346
format bank	美元格式	12.35
format hex	用 16 进制表示	4028b0fcd32f707a
format rat	两个小整数的比	1000/81
format compact	隐藏多余的换行符	
format loose	使用多余的换行符	
format +	只显示这个数的正负	+

所有例子都以 12．345678901234567 为例子默认的格式可以改变格式以显示更多的有效数字，用科学计数法来显示，精确到小数点后两位，显示或隐藏多余的换行符。

2.6.2 disp 函数

另一种显示数据的方法是用 disp 函数。disp 需要一个数组参数，它将值将显示在命令窗口(The Command Windows)中。如果这个数组是字符型（char），那么包含在这个数组中的字符串将会打印在命令窗口(The Command Windows)中。

此函数可联合 num2str(将一个数转化为字符串)和 int2str(将一个整数转化为字符串)来产生新的信息，显示在命令窗口(The Command Windows)中。例如，下面的语句将“the value of pi=3.1416”显示在命令窗口(The Command Windows)中。第一句创建了一个字符型数组,第二句用于显示这个数组。

```
str=['the value of pi='    num2str(pi)];
disp(str);
```

2.6.3 用 fprintf 函数格式化输出数据

用 fprintf 函数显示数据是一种十分简便方法。fprintf 函数显示带有相关文本的一个或多个值，允许程序员控制显示数据的方式。它在命令窗口打印一个数据的一般格式如下：

```
fprint(format,data)
```

其中 format 用于代表一个描述打印数据方式的子符串，data 代表要打印的一个或多个标量或数组。format 包括两方面的内容，一方面是打印内容的文本的提示；另一方面是打印的格式。例如，函数

```
fprintf('The value of pi is %6.2f \n',pi)
```

将会打印出'The value of pi is   3.14',后面带有一个换行符。转义序列%6.2 代表在本函数中的第一个数据项将占有 6 个字符宽度，小数点后有 2 位小数。

fprintf 函数有一个重大的局限性，只能显示复数的实部。当我们的计算结果是复数时，这个局限性将会产生错误。在这种情况下，最好用 disp 显示数据。

表 2.4 fprintf 函数 format 字符中的特殊字符

format string	结果
%d	把值作为整数来处理
%e	用科学记数法来显示数据
%f	用于格式化浮点数，并显示这个数
%g	用科学记数格式，或浮点数格式，根据那个短，并显示之
\n	转到新的一行

例如，下列语句计算复数 x 的值，分别用 fprintf 和 disp 显示

```
x=2*(1-2*i)^3;
str=['disp: x = ' num2str(x)];
disp(str);
fprintf('fprintf: x = %8.4f\n',x);
```

打印的结果如下

```
disp: x = -22+4i
fprintf: x = -22.0000
```

注意 frpintf 忽略了虚部。

编程隐患

fprintf 函数只能复数的实部，所以在有复数参加或产生的计算中，可能产生错误的结果。

2.7 数据文件

有许多的方法用于加载和保存 **MATLAB** 的数据文件，这些方法将会在第八章中介绍。在这里我们只向大家介绍最简章的 save 和 load 命令。

save 命令用于保存当前 **MATLAB** 工作区内的数据到一个硬盘文件。这个命令的基形式如下

```
save filename var1 var2 var3
```

filename 代表你要保存变量的那个文件，var1,var2 等是要保存的变量。在默认情况下，

这个这个文件的扩展名为‘mat’,我们称之为 MAT 文件。如果在 filename 后面无变量，则工作区的所有内容将会被保存。

MATLAB 用一种特殊的复杂形式来存储数据，包括了许许多多的细节，例如变量名和变量类型，数组的大小，以及所有变量值。一个在任何一个平台上创建的 MAT 文件(pc, mac, unix)在另一个平台上都可以应用。它的缺点是 MAT 文件的存储格式不能被其他程序读取。如果一个数据必须由其他程序所读取，那么必须转化为 ASCII 码，并将这些数值写到一个以 ASCII 码为编码的文件中。但是，当以 ASCII 的形式存储，像变量名和变量类型这样的信息就会丢失，产生的数据结果将会更大。

例如，假设数组 x 的定义如下

```
x=[1.23 3.14 6.28; -5.1 7.00 0];
```

命令“save x.dat x -ascii”将会创建一个文件 x.dat，包括数据如下

```
1.2300000e+000    3.1400000e+000    6.2800000e+000
-5.1000000e+000    7.0000000e+000    0.0000000e+000
```

用这种格式定的数据能被其他语言编写的程序或扩展页读取，所以它能帮助 **MATLAB** 程序和其他程序之间共享数据。

好的编程习惯

如果数据需要在 MATLAB 和其他程序之间交换使用，那么以 ASCII 格式存储数据。如果只在 MATLAB 中使用那么，应以 mat 文件的形式存储数据。

MATLAB 并不关心 ASCII 码的扩展名是什么？但是，用户最好用它的传统扩展名“dat”。

好的编程习惯

以“dat”的扩展名保存 ASCII 数据文件，以区别于以“mat”为扩展名的 mat 文件。

Load 命令与 save 命令相反。它从硬盘文件加载数据到 **MATLAB** 当前工作区。这个命令的基本格式为

```
load filename
```

filename 代表所加载文件的文件名。如果这个文件是 mat 文件，那么所有被子加载的变量的变量名的变量类型将和原来一样。如果一个变量包含在工作区间窗口，那么这些数据将会被修复。

MATLAB 能够加载由其他程序创建的 ascii 格式的数据文件。它首先检查所要加载的文件是 mat 文件还是 ascii 文件。如果在 load 语句中加入-ascii 中，则强制 **MATLAB** 把这个文件看作 ASCII 文件。这个文件的内容将会被转化为一个 **MATLAB** 的数组，这个数组名就所要加载的文件名。例如，假设一个名为 x.dat 的 ascii 文件包括下列数据：

```
1.23 3.14 6.28
-5.1  7.00 0
```

那么“load x.dat”将会在当前工作区创建一个 2×3 数组 x,包含数据值。

测试 2.3

本测试提供了一个快速的检查方式，看你是否掌握了 2.6 和 2.7 的基本内容。如果你对本测试有疑问，你可以重读 2.6 和 2.7，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1.如何让 **MATLAB** 显示一个实数，带有十五个有效的数字，并有指数形式？

2.下列语句的作用是什么？它的输出是什么？

(a)
```
radius = input('Enter circle radius:\n');
area = pi * radius^2;
str = ['The area is ' num2str(area)];
```

```
disp(str);
```

(b)
```
value = int2str(pi);
disp(['The value is ' value '!']);
```

3.下列语句有作用是什么？它的输出是什么？

```
value = 123.4567e2;
fprintf('value = %e\n',value);
fprintf('value = %f\n',value);
fprintf('value = %g\n',value);
fprintf('value = %12.4f\n',value);
```

2.8 标量运算和数组运算

在 **MATLAB** 赋值语句中的计算，它的一般形式如下

```
variable_name = expression;
```

赋值语句计算出等号右边表达式的值，然后赋值于等号左边的变量名。注意这个等号并不是传统意义上的等号，它的意义是：存储表达式的值到左边的变量，由于这个原因，等号在这里应叫做赋值号。像

```
ii = ii + 1;
```

这样的语句在数学上是毫无意义的，但在 **MATLAB** 语言中，它有其固有的意义。

它的意义是：把变量 ii 加上 1 之后，再把值存储到变量 ii 中。

2.8.1 标量运算符

位于赋值号右边的表达式，可以包含标量，数组，括号和数学符号的任一个有效联合运算。两标量间的标准运算符号如表 2.5 所示。

当我们需要的时侯，我们可以运用括号来控制运算顺序。括号内的表达式优先于括号外的表达式来计算。例如表达式 2^((8+2)/5)的计算顺序如下

```
2 ^ ( ( 8 + 2 ) / 5 )   = 2 ^ ( 10 / 5 )
                        = 2 ^ 2
                        = 4
```

2.8.2 数组运算和矩阵运算

MATLAB 在数组运算中提供了两种不同类型的运算,一种是数组运算(array operations),一种是矩阵运算(matrix).数组运算是一种用于元素对元素的运算.也就是说,这个运算是针对两数组相对应的运算使用的.例如,$a=\begin{bmatrix}1 & 3\\ 2 & 4\end{bmatrix}$ $b=\begin{bmatrix}-1 & 3\\ -2 & 1\end{bmatrix}$,那么 $a+b=\begin{bmatrix}0 & 6\\ 0 & 5\end{bmatrix}$.注意两数组的行与列必须相同.否则,**MATLAB** 将产生错误.

数组运算可以用于数组与标量的运算.当一个数组和一个标量进行运算时,标量将会和数组中的每一元素进行运算.例如

$a=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$,则 $a+4=\begin{bmatrix}5 & 6\\ 7 & 8\end{bmatrix}$

相对地,矩阵运算则遵守线性代数的一般规则,像矩阵的乘法.在线性代数中,c=a×b 的定

义如下:

表 2.5 两标量间的数学运算符

运算符	代数形式	MATLAB 形式
加号	A+B	A+B
减号	A- B	A- B
乘号	A×B	A*B
除号	$\frac{A}{B}$	A/B
指数	A^B	A^B

$$c(i,j)=\sum_{k=1}^{n}a(i,k)b(k,j)$$

例如 $a=\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$, $a=\begin{bmatrix}-1 & 3\\-2 & 1\end{bmatrix}$,那么 $a\times b=\begin{bmatrix}-7 & -6\\-10 & 10\end{bmatrix}$。注意，在矩阵相乘中，a 阵的列数必须等于 b 阵的行数。

MATLAB 用一个特殊的符号来区分矩阵运算和数组运算。在需要区分两者不同的时侯，把点置于符号前来指示这是一个数组运算（例如，.*）。表 2。6 给出的是一些常见的数组和矩阵运算。

表 2.6 常见的数组和矩阵运算

运算	MATLAB 形式	注释
数组加法	A+B	数组加法和矩阵加法相同
数组减法	A- B	数组减法和矩阵减法相同
数组乘法	A.*B	A 和 B 的元素逐个对应相乘.两数组之间必须有相同的形,或其中一个是标量.
矩阵乘法	A*B	A 和 B 的矩阵乘法.A 的列数必须和 B 的行数相同.
数组右除法	A./B	A 和 B 的元素逐个对应相除: A(i,j)/B(i,j)两数组之间必须有相同的形,或其中一个是标量.
数组左除法	A.\B	A 和 B 的元素逐个对应相除: B(i,j)/A(i,j)两数组之间必须有相同的形,或其中一个是标量.
矩阵右除法	A/B	矩阵除法,等价于 A*inv(B), inv(B)是 B 的逆阵
矩阵左除法	A\B	矩阵除法,等价于 inv(B)*A, inv(A)是 A 的逆阵
数组指数运算	A.^B	AB 中的元素逐个进行如下运算 A(i,j)^B(i,j), A(i,j)/B(i,j)两数组之间必须有相同的形,或其中一个是标量.

初学者往往混淆数组运算和矩阵运算.在一些情况下,两者相互替换会导致非法操作,**MATLAB** 将会报告产生了错误。在另一些情况下,两种运算都是合法的,那么这时 **MATLAB** 进行错误的运算，并产生错误的结果。当我们进行方阵运算时，极易产生这样的错误。两个方阵具有相同的大小,两者之间的数组运算和矩阵运算都是合法的,但产生的结果完全不同。在这种情况下，你要万分的小心。

编程隐患

在你的 MATLAB 代码中，仔细区分数组运算和矩阵运算。数组乘法和矩阵乘法极易混淆。

例 2.1

假设 a,b,c 和 d 的定义如下

$a=\begin{bmatrix}1 & 0\\ 2 & 1\end{bmatrix}$ $b=\begin{bmatrix}-1 & 2\\ 0 & 1\end{bmatrix}$ $c=\begin{bmatrix}3\\ 2\end{bmatrix}$ $d=5$

分别指出下列表达式的运算结果

(a) a + b (b) a .* c (c) a * b (d) a * c
(e) a + c (f) a + d (g) a .* d (h) a * d

答案：

（a） 这是一个数组或矩阵加法：$a+b=\begin{bmatrix}0 & 2\\ 2 & 2\end{bmatrix}$。

（b） 这是一个数组乘法：$a.*b=\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}$。

（c） 这是一个矩阵乘法：$a*b=\begin{bmatrix}-1 & 2\\ -2 & 5\end{bmatrix}$。

（d） 这是一个矩阵乘法：$a*c=\begin{bmatrix}3\\ 8\end{bmatrix}$。

（e） 操作非法，两数组形不同

（f） 数组与标量的加法：$a+d=\begin{bmatrix}6 & 5\\ 7 & 6\end{bmatrix}$。

（g） 数组乘法：$a.*d=\begin{bmatrix}5 & 0\\ 10 & 5\end{bmatrix}$。

（h） 矩阵乘法：$a*d=\begin{bmatrix}5 & 0\\ 10 & 5\end{bmatrix}$。

矩阵的左除运算有着十分重要意义，我们必须理解它。一个 3×3 的线性方程组的形式如下

$$\begin{aligned}a_{11}x_1+a_{12}x_2+a_{13}x_3&=b_1\\ a_{21}x_1+a_{22}x_2+a_{23}x_3&=b_2\\ a_{31}x_1+a_{32}x_2+a_{33}x_3&=b_3\end{aligned} \tag{2.1}$$

可以写成如下形式

$$Ax=B \tag{2.2}$$

$$A=\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}，\quad A=\begin{bmatrix}b_1\\ b_2\\ b_3\end{bmatrix} \text{和} x=\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}$$

2.2 式的结果可以用线性代数的知识来解决。结果是

$$x=A^{-1}B \tag{2.3}$$

因为左除运算 A\B 等价于 inv(A)*B，所以左除运算是解线性方程的好方法。

2.9 运算的优先级

许多的数学运算写入一个表达式是非常平常的事。例如，考虑初速度为 0 的匀加速运动的位移表达式

```
distance = 0.5 * accel * time ^ 2
```

这个表达式有二个乘法运算和一个幂运算。在这样的表达式中，知道运算的先后顺序是十分重要的。如果幂运算先于乘法运算执行，这个表达式等价于

```
distance = 0.5 * accel * (time ^ 2)
```

如果乘法运算先于幂运算执行，这个表达式等价于

```
distance = (0.5 * accel * time) ^ 2
```

这两个式子将产生不同的结果，所以我们必须清楚它们中那个是正确的。

为了使表达的值精确，**MATLAB** 建立了一系列的规则控制运算的层次或顺序。这些规则一般情况下遵循代数的运算法则。数学运算的顺序如表 2.7。

表 2.7 运算的优先级

优先级	运算
1	括号里的内容先运算，从最里面的括号去运算
2	幂运算，从左向右
3	乘除法，从左向右
4	加减法，从左向右

例 2.2

变量 a,b,c,d 初始化如下

a = 3; b = 2; c = 5; d = 3;

计算如下的 **MATLAB** 的赋值语句

(a) output = a*b+c*d;

(b) output = a*(b+c)*d;

(c) output = (a*b)+(c*d);

(d) output = a^b^d;

(e) output = a^(b^d);

正如我们看到的，运算的顺序对一个代数表达式的最终值产生重大的影响。

将程序中的每个表达式尽量写清楚，这是十分重要的。编写的程序不仅要能够计算出所要求的值的来，在需要的时侯，还要考虑它的可维护性。你应当经常问自己“六个月后我能看得懂我现在编得程序吗？其他的程序员看到我的代码，他能迅速的理解吗？”。如果在你的心中有所疑虑，那就用更多的括号使之更加清晰。

好的编程习惯
在需要的时侯用括号使用表达式更加清晰和易于理解。

如果在一个表达式中用到括号，那么括号必须平衡。也就是说，左括号数与右括号数相等。如果两者数目不相同，那么将会导致错误的产生。这种错误经常在输入过程中发生，当 **MATLAB** 编译器在执行这个命令时被发现。例如

```
（2 + 4) / 2)
```

在执行时将会出现一个错误。

测试 2.4

本测试提供了一个快速的检查方式，看你是否掌握了 2.8 和 2.9 的基本内容。如果你对本测试有疑问，你可以重读 2.8 和 2.9，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1.假设 abcd 的定义如下，计算下面合法运算的结果，如果不合法，指出原因

$$a=\begin{bmatrix}2 & 1\\ -1 & 2\end{bmatrix}\qquad b=\begin{bmatrix}0 & -1\\ 3 & 1\end{bmatrix}\qquad c=\begin{bmatrix}1\\ 2\end{bmatrix}\qquad d=-3$$

(a) result = a .* c;
(b) result = a * [c c];
(b) result = a .* [c c];
(d) result = a + b * c;
(e) result = a + b .* c;

2.求矩阵 x，已知 $Ax=B$,

$$A=\begin{bmatrix}1 & 2 & 1\\ 2 & 3 & 2\\ -1 & 0 & 1\end{bmatrix}\qquad B=\begin{bmatrix}1\\ 1\\ 0\end{bmatrix}$$

2.10 MATLAB 的内建函数

2.10.1 选择性结果

与数学的函数不同，**MATLAB** 函数返回一个或多个值给调用函数。max 函数就是这样的一个例子。这个函数一般情况下返回输入向量中的最大值，但是它返回的第二个参数是输入向量中的最大值在向量中的位置。例如，语句

```
maxval = max ([1 -5 6 -3])
```

返回的结果为 maxval=6,但是要有两个返回值，那么这个函数包括最大所处的位置。

```
[maxval index] = max ([1 -5 6 -3])
```

将会产生结果 maxval=6,和 index=3.

2.10.2 带数组输入的 MATLAB 函数的应用

许多 **MATLAB** 函数定义了一个或多个标量输入，产生一个输出。例如，语句 y=sin(x) 计算了 x 的正弦，并将结果存储到 y 变量中。如果这些函数接受了输入值构成的数组，那么 **MATLAB** 将一一计算出每个元素所对应的值。例子，假设

```
x=[0 pi/2 3*pi/2 2*pi]
```

那么语句

```
y=sin(x)
```

将会产生 y=[0 1 -1 -0].

2.10.3 常见的 MATLAB 函数

一些极其常用的**MATLAB**函数列入了表2.8中.这些函数将会用在以后的例子的作业中.如果你要加载不在下表中的函数,那么你通过前面介绍的方法,搜索适当的函数.

注意与大多数的计算语言不同,许多的 **MATLAB** 函数能够正确计算出复数结果.**MATLAB** 自动计算出正确的结果,尽管其结果可能是虚数和复数.例如,在 C 和 Fortan 语言中运行函数 sqrt(-2)时将会出现运行时错误.相反地,**MATLAB** 将会产生虚部答案.

```
>> sqrt(-2)
ans =
        0 + 1.4142i
```

2.11 画图入门

MATLAB 的扩展性和机制独立的画图功能是一个极其重要的功能.这个功能使数据画图变得十分简单.画一个数据图,首先要创建两个向量，由 x, y 构成,然后使用 plot 函数.

例如,假设我们要画出函数 $y=x^2$-10x+10 的图象,定义域为［0,10］.只需要 3 个语句就可以画出此图.第二句用于计算 y 值(注意我们用的是数组运算符,所以可以对 x 的元素一一运算.).最后打印出此图.

```
x = 0:1:10;
y = x.^2-10*x+15;
plot(x,y);
```

当执行到 plot 函时,**MATLAB** 调用图象窗口,并显示图象.如图图 2.4.

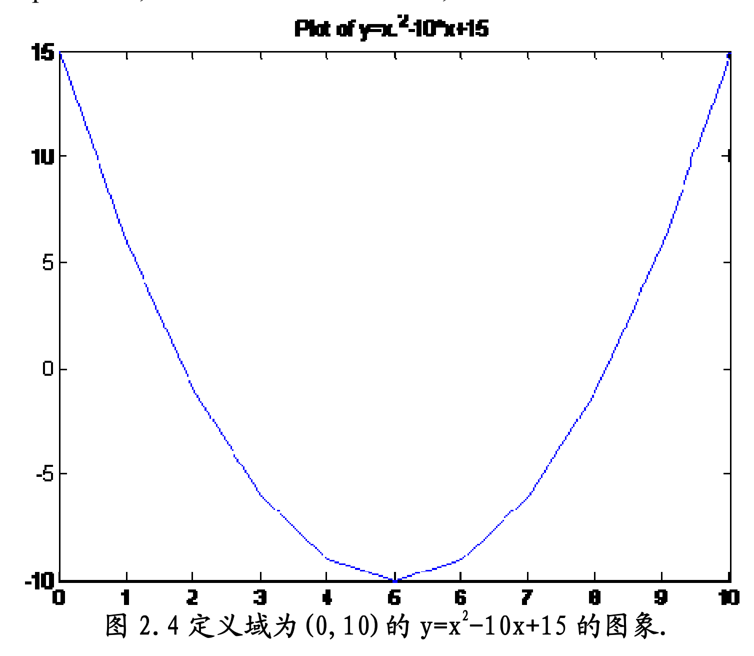


图 2.4 定义域为(0,10)的 $y=x^2-10x+15$ 的图象.

表 2.8 常见的 **MATLAB** 函数

函数	描述
数学函数	
abs(x)	计算 x 的绝对值
acos(x)	计算 x 的反余弦函数

表 2.8 常见的 **MATLAB** 函数

函数	描述
数学函数	
angle(x)	计算复数 x 的幅角
asin(x)	计算 x 的反正弦函数值
atan(x)	计算 x 的反正切函数值
atan2(y,x)	$\tan^{-2}(y/x)$
cos(x)	cosx
exp(x)	e^x
log(x)	$\log_e x$
[value,index]=max(x)	返回 x 中的最大值，和它所处的位置
[value,index]=min(x)	返回 x 中的最小值，和它所处的位置
mod(x,y)	余数
sin(x)	sinx
sqrt(x)	x 的平方根
tan(x)	tanx
rounding(取整)函数	
ceil(x)	
fix(x)	
round(x)	
字符转换函数	
char(x)	将矩阵中的数转化为字符,矩阵中的元素就不大于 127
double(x)	将子符串转化为矩阵
int2str(x)	将整数 x 转化为字符串形式
num2str(x)	将带小数点的数转化为一个字符型数组
str2num(x)	将字符串转化为数

2.11.1 简单的 xy 画图

正如我们所看到的,在 **MATLAB** 中画图是十分容易的.只要任何一对向量的长度相同,那么它就可以就能可视化地画出来.但是这还不是最后的结果,因为它还没有标题,坐标轴标签,网格线.

给图增加标题和坐标轴标签将会用到 title, xlabel, ylable 函数。调用每个函数时将会有一个字符串，这个字符串包含了图象标题和坐标轴标签的信息。用 grid 命令可使网格线出现或消失在图象中，grid on 代表在图象中出现网格线，grid off 代表去除网格线。例如下面的语句将会产生带有标题，标签和网格线的函数图象。结果如图 2.5 所示。

```
x=0:1:10;
y=x.^2-10*x+15;
plot(x,y);
title ('Plot of y=x.^2-10*x+15');
xlabel ('x');
ylabel ('y');
grid on;
```

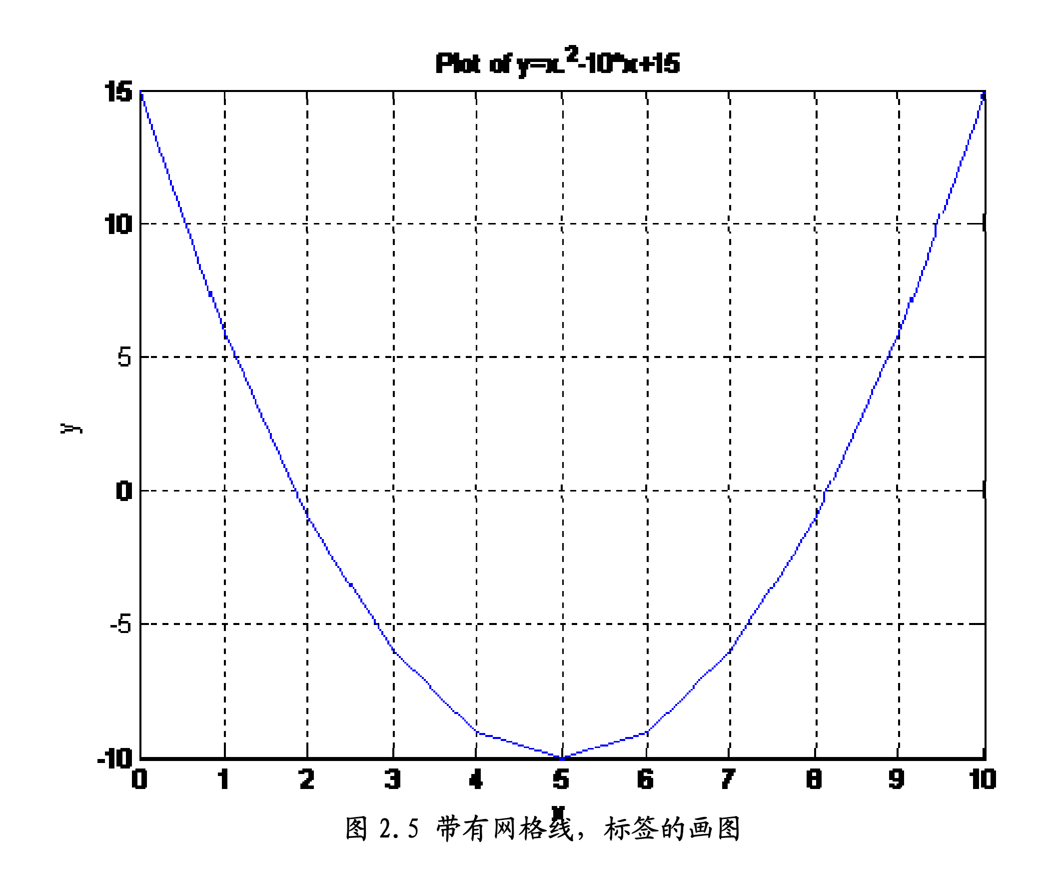


图 2.5 带有网格线，标签的画图

2.11.2 打印图象

一个图象一旦建立，我们就可以用 print 命令在打印机上打印出这幅图，也可以单击图象窗口的打印图标或者在文件菜单中选择打印项打印。

print 命令的一般形式如下：

```
print <选项> <文件名>
```

如果没有文件名，这个命令就会命令打印机打印当前图片。如果带有文件名，那么这个命令就会打印这个图片到指定的文件。有许多的选项指定输出到文件或打印机的格式。一个最重要的选项是-dtiff.这个选项指定输出图片的格式是标签影像档案格式（TIFF）。因为在 PC，Mac 和 UNIX 平台上的文字处理软件都支持这种格式。这就使得在文档中插入 **MATLAB** 图象变得十分的简单。下面这个命令将会创建一个 TIFF 格式的当前图象的图片，并保存在一个叫 my_image.tif 的文件中

```
print –dtiff my_image.tif
```

你也可以选择图象窗口中的“file/export”选项来创建 tiff 图片。

2.11.3 联合作图

在同一坐标内作出多个函数的图象的情况是十分常见的。假如，你要在同一坐标轴内作出 f(x)=sin2x 和他的微分函数的图象。它的微分式为

$$\frac{d}{dt}\sin 2x = 2\cos 2x \qquad (2.4)$$

在同一坐标系内打印两个函数，我们必须产生一系列的 x 值和每一个函数分别对应的 y 值。然后利用这些值画出图象，plot 函数的格式如下所示：

```
x = 0:pi/100:2*pi;
y1 = sin(2*x);
y2 = 2*cos(2*x);
plot (x,y1,x,y2);
```

所得图像如图图 2.6 所示。

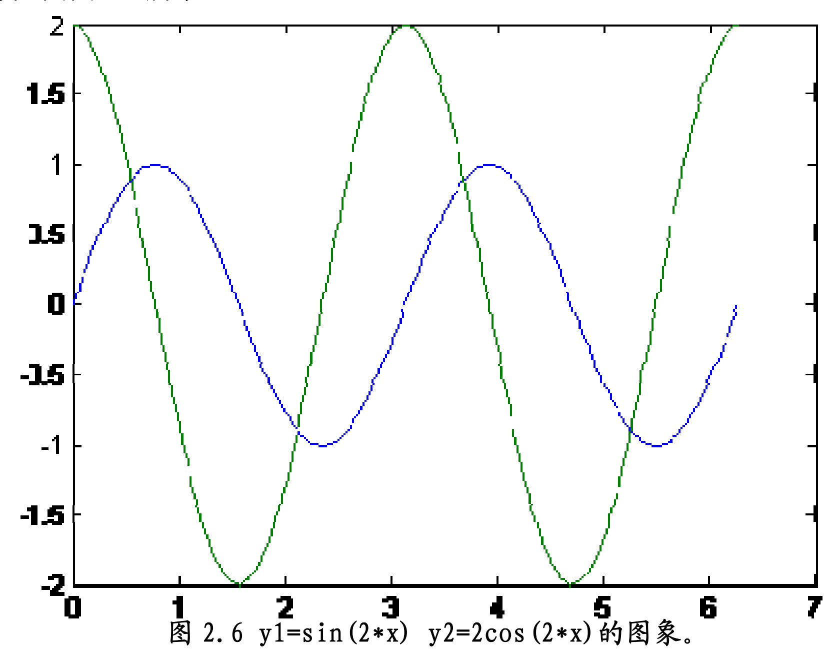


图 2.6 y1=sin(2*x) y2=2cos(2*x)的图象。

2.11.4 线的颜色,线的形式,符号形式和图例

MATLAB 允许程序员选择轨迹的颜色,轨迹的形式,和符号的类型.在 X,Y 向量参数后带有这些属性的字符串的 plot 函数,可以选择这些细节.

这些属性字符串包括三个方面,

第一方面指定轨迹的颜色,

第二方面指定符号的类型,

第三方面指定线的类型.

各种颜色,符号和线的类型将在表 2.9 中显示.

表 2.9 图象的颜色，标记（符号）类型，线型

颜色		标记类型		线型	
y	黄色	.	点	-	实线
m	品红色	o	圈	:	点线
c	青绿色	x	×号	-.	画点线
r	红色	s	正方形	--	虚线
g	绿色	d	菱形	<none>	无
b	蓝色	v	倒三角		
w	白色	^	正三角		
k	黑色	>	三角(向右)		

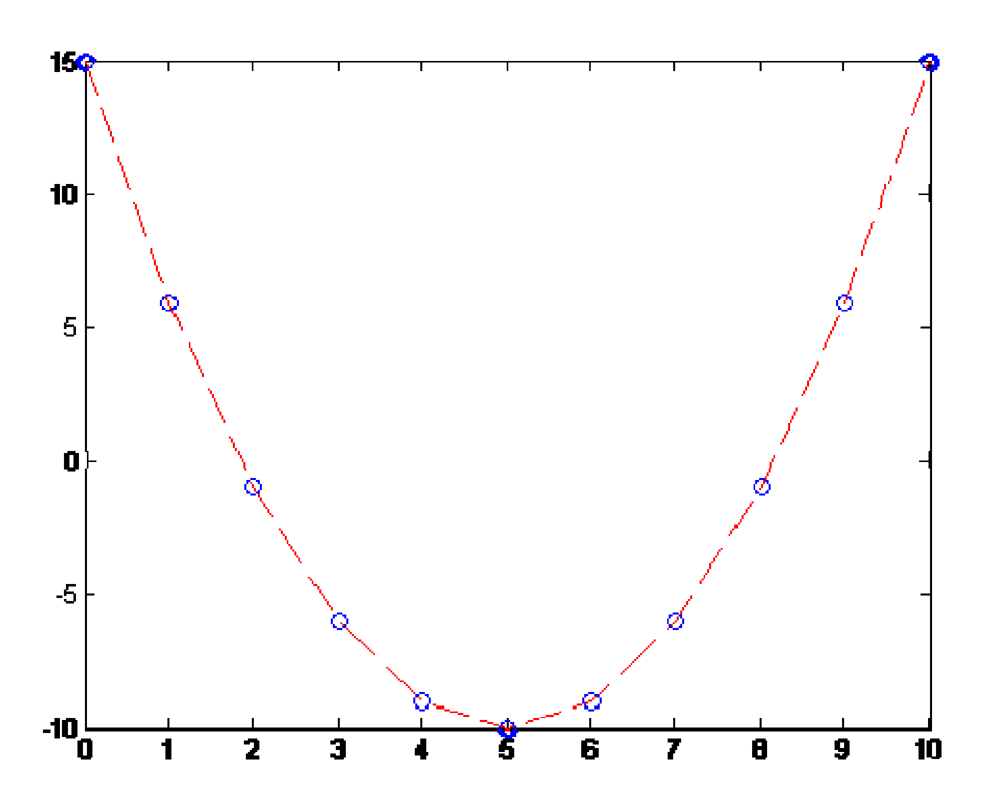
表 2.9 图象的颜色，标记（符号）类型，线型

颜色		标记类型		线型	
		<	三角(向左)		
		p	五角星		
		h	六线形		
		<none>	无		

这些属性字符串可以任意的混合使用.如果有多个函数,每个函数都有它自己的属性字符串.

例如,函数 $y=x^2-10x+15$ 的图象,曲线为红色的虚线,重要的数值用蓝色的小圆圈表示.

```
x = 0:1:10;
y = x.^2 -10.*x +15;
plot(x,y,'r--',x,y,'bo');
```



我们可以用 legend 来制作图例。它的基本的形式如下

```
legend('string1','string2',...,pos)
```

其中 string1,string2 等等是与轨迹标签名，而 pos 是一个整数，用来指定图例的位置。这些整数所代表的意义在表 2.10 中的列出。用 legend off 命令将能去除多余的图例。一个完整的图象例子将会显示图 2.7 中，产生这个图象的语句如下所示。图 2.7 在同一坐标系内，显示了 f(x)=sin2x 和它的微分函数的图象，用黑实线代表 f(x),用红虚线代表它的微分函数。图中有标题，坐标轴标签和网格线。

```
x=0:pi/100:2*pi;
y1=sin(2*x);
y2=2*cos(2*x);
plot(x,y1,'k-',x,y2,'b--');
title(' Plot of f(x)=sin(2x) and its derivative');
xlabel('x');
ylabel('y');
legend('f(x)','d/dx f(x)')
grid on;
```

2.11.5 对数尺度

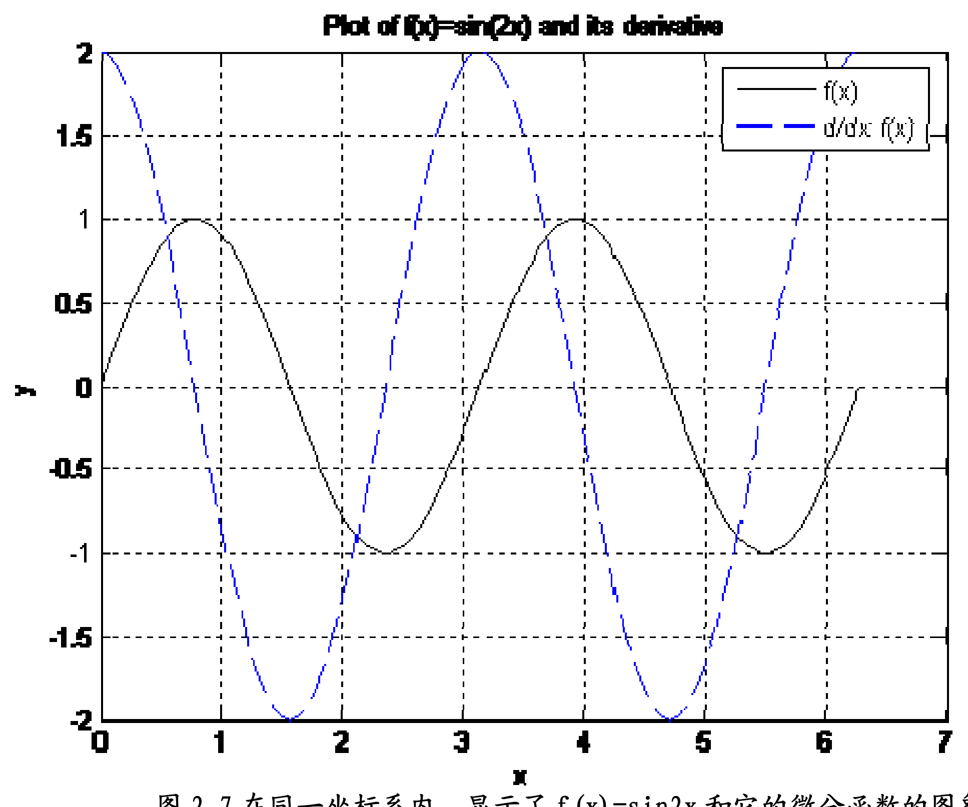


图 2.7 在同一坐标系内，显示了 f(x)=sin2x 和它的微分函数的图象

打印数据既可以用对数尺度，也可以用线性尺度。在 x,y 轴上使用这两种尺度的一种或两种可以组合形成 4 种不同的坐标系。每一种组合者有一个特定的函数。

1.plot 函数的 x,y 均用线性尺度

2.semilog 函数 x 轴用对数尺度，y 轴将用线性尺度

3.semiloge 函数 x 轴用线性尺度，y 轴用对数尺度

4.loglog 函数两坐标轴将会都用对数尺度。

这四个函数在意义上是等价的，只是坐标轴的类型不同。每一个图象的例子如图 2.8 所示。

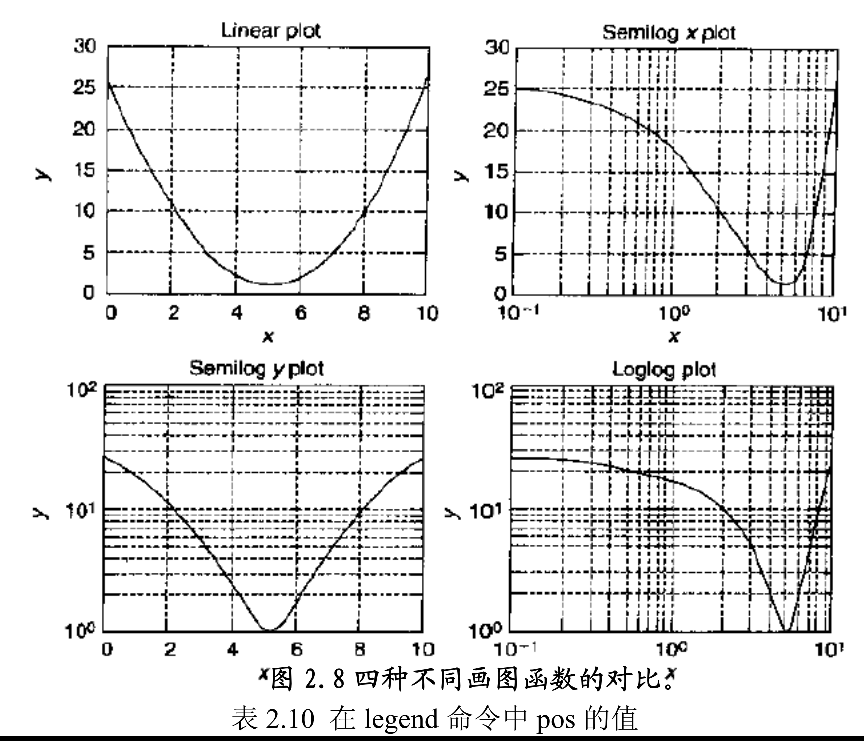


图 2.8 四种不同画图函数的对比

表 2.10 在 legend 命令中 pos 的值

值	意义
0	自动寻找最佳位置，至少不与数据冲突
1	在图象的右上角
2	在图象的左上角
3	在图象的左下角
4	在图象的右下角
-1	在图象的右边

2.12 例子

下面的例子将向大家介绍如何用 **MATLAB** 解决问题。

例 2.3

（温度转换）设计一个 **MATLAB** 程序，读取一个华氏温度的输入，输出开尔文温度。
答案
华氏温度和开尔文温度的转换关系式可在物理学课本中找到。其关系式为：

$$T(\text{开尔文}) = \left(\frac{5}{9}T(\text{摄氏度}) - 32.0\right) + 273.15 \qquad (2.5)$$

在物理学参考书中举了一些例子，我们可以用来检验我们程序是否正确。例如

	华氏度（^{o}C）	开尔文(K)
沸水的温度	212	373.15
冰水混合物的温度	-110	194.26

我们设计程序的步骤如下
1.提示用户键入华氏温度值
2.读取输入值

3.通过关系式转换为开氏温度
4.输出结果，结束
我们将会用 input 函数输入华氏温度，用 fprintf 函数输出结果。

```
%    Script file:temp_conversion.m
%
%    Purpose:
%    To convert an input temperature from degrees Fahrenheit to
%    an output temperature in kelvins.
%
%    Record of revisions:
%    Date        Programmer     Description of change
%    ====        =========      ================
%    12/01/97    S.J.Chapman    Original code
%
%Define variables:
%    temp_f    --Temperature in degrees Fahrenheit
%    temp_k    --Temperature in kelvins

%Prompt the user for the input temperature.
temp_f=input('Enter the temperature in degrees Fahrenheit:');
%Converttokelvins.
temp_k=(5/9)*(temp_f-32)+273.15;
%Writeouttheresult.
fprintf('%6.2f degrees Fahrenheit = %6.2f kelvins.\n',...
temp_f,temp_k);
```

我们输入上面的例子中的华氏温度值，以检测程序的正确性。注意用户的输入值已用黑体字标出。

```
>> temp_conversion
Enter the temperature in degrees Fahrenheit:212
212.00 degrees Fahrenheit = 373.15 kelvins.
>> temp_conversion
Enter the temperature in degrees Fahrenheit:-110
-110.00 degrees Fahrenheit = 194.26 kelvins.
```

这个结果和物理教科书的结果相同。在本程序中,我们重复出带单位的输入值和输出值.只有带上单们神经质输出才有意义.

按照惯例,任何输入变量和输出变量的单位都应打印出来.

好的编程习惯
当你读取和写入数据时,使用适当的单位

例 2.4

电子工程：负载的最大输出功率

一个内阻 R_s＝50Ω，电动势 V＝120V 的电源驱动一个负载 R_L。当 R_L 为多少时，R_L 的功率最大？在这种情况下，功率为多少？画以 R_L 为自变量的 R_L 功率图。

答案：

在本程序中，我们需要改变 R_L 的值，然后计算出每一个 R_L 的功率。R_L 功率的表达式为

$$P_L = I^2 R_L \tag{2.6}$$

I 代表流经负载的电流。电流可由欧姆定律计算得到。

$$I = \frac{V}{R_{总}} = \frac{V}{R_s + R_L} \tag{2.7}$$

这个问题解决的步骤如下

1. 创建一个数组。这个数组是以 1 为起始项，以 1 步长的等差数组，共 100 项，这是 R_L 的取值。
2. 计算 R_L 的电流，
3. 计算每个 R_L 的功率。
4. 画出 R_L 的功率图，以确定 R_L 为多少时其功率最大。

整个程序的代码如下：

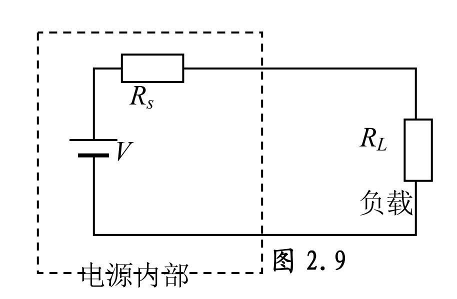


图 2.9

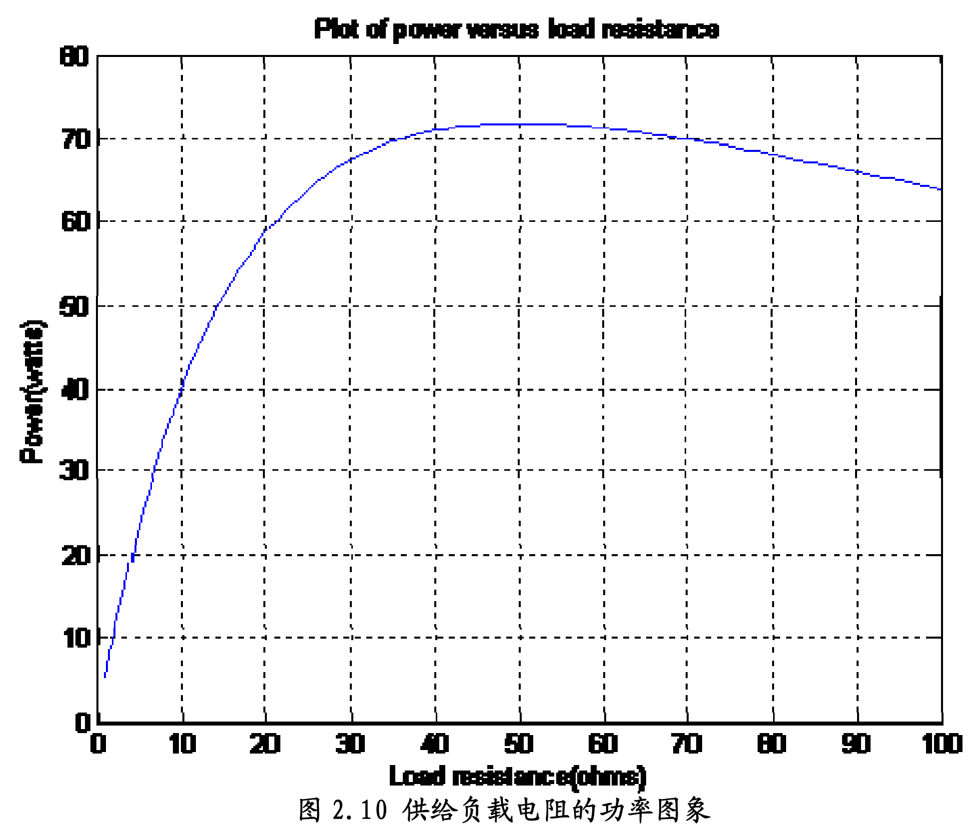


图 2.10 供给负载电阻的功率图象

```
%   Script file:calc_power.m
%
%   Purpose:
%   To calculate and plot the power supplied to a load as
%   a function of the load resistance.
%
%   Record of revisions:
%   Date  Programmer   Descriptionofchange
%   ====        ==========   =====================
%   12/01/98 S.J.Chapman Original code
%
%Define variables:
```

```
%    amps      --Current flow to load(amps)
%    pl    --Power supplied to load(watts)
%    rl    --Resistance of the load(ohms)
%    rs    --Internal resistance of the power source(ohms)
%    volts--Voltage of the power source(volts)
%Set the values of source voltage and internal resistance
volts=120;
rs=50;
%Create an array of load resistances
rl=1:1:100;
%Calculate the current flow for each resistance
amps=volts./(rs+rl);
%Calculate the power supplied to the load
pl=(amps.^2).*rl;
%Plot the power versus load resistance
plot(rl,pl);
title('Plot of power versus load resistance');
xlabel('Load resistance(ohms)');
ylabel('Power(watts)');
grid on;
```

当这个程序运行时，产生的图象如图 2.10。从这个图我们可知当负载电阻为 50Ω时，功率最大。最大功率为 72W。

注意在本例中，用的是数组运算符.*,.^和./.这些运算符将会使数组 amps 和 pl 按元素一一对应计算.

例 2.5

用 C-14 确定年代

一个元素的放射性同位素是不稳定元素的一种特殊形态.在一段时间内,它会自然的衰变为另一种元素.衰变一种呈指数下降的过程.如果 Q_0 是放射性物质在 t=0 时的初始量,那么它的质量与变量 t 的关系式为

$$Q(t) = Q_0 e^{-\lambda t} \tag{2.8}$$

其中 λ 代表衰变率.

因为放射性元素的衰变是以一定的速率发生的，我们可以把它当作一个时钟来测定的衰变开始的时间。如果我们知道衰变开始时物质的质量和现在放射性元素剩余的质量,我们可以根据公式(2.8)换算出衰变时间 t,即

$$t_{\text{decay}} = -\frac{1}{\lambda}\log_e \frac{Q}{Q_0} \tag{2.9}$$

公式 2.9 在科学的许多领域有着广泛的应用.例如,考古学家可以根据 C14 的衰变周期,来确定古生物距今生活的年代.现在活着的生物 C14 的含量是不变的,所以可以根据古生物 C14 的现存量来确定古生物的生存年代.已知 C14 的衰变率 λ 为 0.00012097/年,所以如果 C14 的剩余量可以通过测量得到,那么我们就可以根据公式 2.9 算出这个生物活在多少年之前.图 2.1 向大家展示了以时间为自变量的 C14 的剩余量函数.

编定一个程序,读取样品中 C14 剩余量的百分比,计算样品距今的年代,并打印出结果.

这个问题解决的步骤如下

1. 提示用户输入样品中 C14 的剩余量
2. 读取百分比

3. 将百分比转化成分数 $\frac{Q}{Q_0}$.
4. 利用公式(2.9)计算出距今的年数
5. 输出结果,结束.

代码如下

```
%    Script file:c14_date.m
%
%    Purpose:
%    To calculate the age of an organic sample from the percentage
%    of the original carbon 14 remaining in the sample.
%
%    Record of revisions:
%    Date Programmer    Description of change
%    ====      ==========  =====================
%    12/02/97  S.J.Chapman   Original code
%
%Define variables:
%     age              --The age of the sample in years
%     lamda            --the radioactive decay constant for carbon-14,in units of 1/years.
%     percent          --The age of carbon 14 remaining at the time of the measurement
%     ratio            --The ratio of the carbon 14 remaining at the time of the measurement to
the original amount of carbon 14.
%Set decay constant for carbon-14
lamda=0.00012097;
%Prompt the user for the percentage of C-14 remaining.
percent=input('Enter the percentage of carbon 14 remaining:\n');
%Perform calculations
ratio=percent/100;    %Convert to fractional ratio
age=(-1.0/lamda)*log(ratio);%Get age in years
%Tell the user about the age of the sample.
string=['The age of the sample is ' num2str(age) 'years.'];
disp(string);
```

我们通过计算 C14 的半周期来来测试这个程序.

测试结果如下

```
>> c14_date
Enter the percentage of carbon 14 remaining:
50
The age of the sample is 5729.9097years.
```

在化学物理 CRC 手册(The CRC Handbook of Chemistry and Physics)中,C14 的半衰期为 5730 年.我们计算的结果和参考书相符.

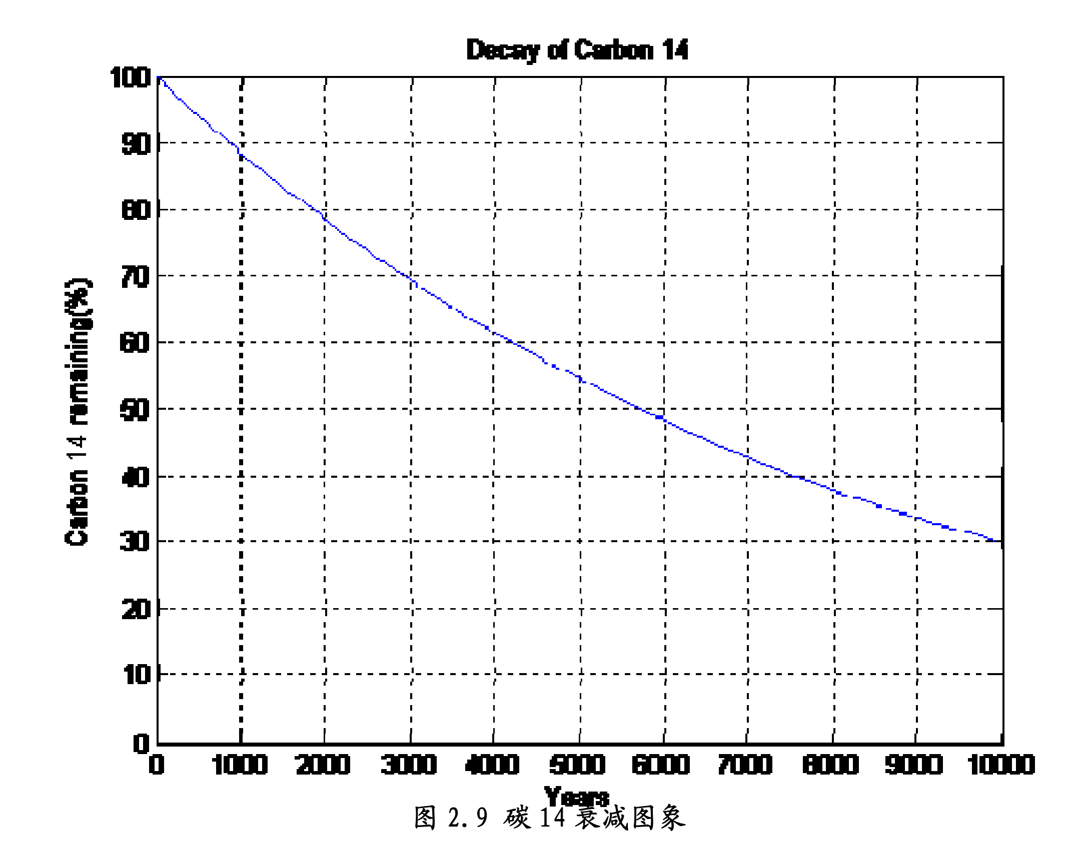


图 2.9 碳 14 衰减图象

2.13 调试 MATLAB 程序

有一个古老的说法,人这一生唯一能够确定的东西是死亡和税收.我们在这里再增加一项,无论你编定多大的程序,你第一次运行时,肯定通不过!程序中的错误我们称之为 BUGS,找出并排出它们,我们称之为调试(debugging).已知一个程序,而无法运行,我们怎样调试它呢?

在 **MATLAB** 中有三种类型的错误.第一种错误是**语法错误**.语法错误是 **MATLAB** 自身的错误,例如拼写错误和标点错误.当编译 M 文件时,maltab 编译器将会找出这些错误.例如,语句

```
x = ( y + 3 ) / 2 );
```

有一个语法错误,因为其括号不平衡.

如果这句存储在 M 文件 test.m 中，当 test 编译进，将会出现下面的信息。

```
>> test
??? Error: File: d:\MATLAB7\work\test.m Line: 1 Column: 10
Unbalanced or misused parentheses or brackets.
```

第二种类型的错误是一种运行时错误。当一个非法的数学运算出现在程序的过程(例如,除以 0)，将会出现运行时错误。这些错误将会使程序返回 Inf 或 NaN,用来参与下一步的运算。导致这个程序的结果变无效。

错误的第三种形式是**逻辑错误**(logical error).逻辑错误是指编译和运行都能通过，而产生了错误的结果。

在编程过程中出现的最普遍的错误是书写错误。一些书写错误可能产生无效的 **MATLAB** 语句。这些错误产生的语法错误可能会被编译器发现。另一个书写错误发生在变量名的书写上。例如，变量中的字符可能被调换，漏写或错写。这样就会创建一个新的变量，在前面我们已经提到，**MATLAB** 能够很容易地创造一个新的变量，它不会发现这个错误。书写错误也能导致逻辑错误。例如，如果变量 vel1 和 vel2 都在程序中代表速度，如果一时疏忽用其中一个替代了另一个，那么你就只能用人工检查代码找出此类错误。有的时侯程序

开始时能够执行，但是运行时错误和逻辑错误可能在执行中发生。在这种情况下，可能是输入错误，也可能是逻辑结构错误。找出这类错误的第一步是检查程序的输入数据。既可以去掉输入语句后的分号，也可以加入一个多余的输出语句以证明这个输入值是不是你想要的。如果你已经排除了变量名错误和输入值错误，接着你要处理的是逻辑错误。你应该检测是否有逻辑错误，应当检查每一个赋值语句。

1.如果一个赋值语句非常的长，把他分成许多小的赋值语句。小的语句易证明。

2.检查你的赋值语句中括号的放置。在赋值语句中，由于括号导致运算顺序错误是极其常见的错误。如果你对运算顺序仍有疑问，应该多加括号，使之更加清晰。

3.保证每个变量正确的初始化。

4.保证函数中用到的单位统一。例如，在三角函数中输入必须是弧度值，而不是角度值。如果你仍然得到的是错误的语句，在更多的位加上输出语句，以检查中间计算。如果你能确定错误的位置，那么你就知道在那里找到问题所在，百分九十五地在这片区域内。如果问题依然存在，那么这时你就应当把你遇到的问题解释给你的同学或老师，让他们给你检查错误。一个人看自己编写的代码找不到错误是非常常见的，而其他的人则可以迅速地找出错误的地方，而这个地方你可能已经看了一次又一次。

好的编程习惯
确保你在编程设计过程：
1.初始化所有变量
2.适当应用括号使运算顺序清晰以减少调试的工作量

在 **MATLAB** 中有一个专门的调试器，叫做 symbolic debugger. symbolic debugger 允许用户一句一句地执行语句，检测出所有的变量值，它能让你看到所有的中间值，而不用在其中加入输出语句。我们将会在第三章中介绍 symbolic debugger。

2.14 总结

在本章中，我们将向大家介绍了两种数据类型：double 和 char.我们还向大家介绍了赋值语句，数学计算，常用函数,输入输出语句和数据文件.

MATLAB 表达的运算顺序遵守一定的规则,即优先级高的先执行,优先级低的后执行.运算的优先级总结在表 2.11 中.

表 2.11 运算的优先级

优先级	运算
1	括号里的内容先运算，从最里面的括号去运算
2	幂运算，从左向右
3	乘除法，从左向右
4	加减法，从左向右

MATLAB 语言包括许许多多地的内建函数,帮助我们解决问题.它的函数比 C 和 Fortan 语言中的函数要多得多,包括机制独立的画图功能.一些常见的固有函数在表 2.8 中列出,其他函数将会在以后的章节中介绍.所有 **MATLAB** 的函数列表可在在线帮助浏览窗口中得到.

2.14.1 好的编程习惯总结

每一个 **MATLAB** 程序都应让其他熟悉 **MATLAB** 编程的人容易理解.所以有一个好的编程习惯十分重要,因为它能使一个程序使用很好时间.过了一个段时间,条件可能改变,程序也可能要改变以适应这些变化.修改这个程序的人可能是其他人而不是这个程序的原作者.这个

程序员在修改程序之前必须先理解原程序.

编写清晰,易理解,可维护强的程序要比编写简单的程序要难得多.一个程序员必须发展这方面的能力以证明自己的工作,还有程序必须避免一些常见的错误.下面的指导意见,将有助于你养成好的编程习惯.

1. 尽可能的使用有意义的变量名,一眼就可以看懂,像 day,month,year.
2. 给每一个程序创建一个数据字典,以提高程序的可维护性.
3. 变量名一律用小写字母,这样可以不会因大小写不同而造成变量混淆。
4. 在所有的 **MATLAB** 赋值语句的后面加上一个分号,用来禁止赋值的重复.在程序调试期间,如果你检验检某个语句的值,可去掉语句后的分号.
5. 如果要在 **MATLAB** 和其他程序之间交换数据,那么就要以 ASCII 格式存储数据.如果数据只应用在此 **MATLAB** 中那么,应以 mat- file 格式存储数据.
6. 以”dat”为扩展名保存 ASCII 数据以区分 MAT 文件,MAT 文件的扩展名为 mat.
7. 用适当的括号使你的表达式清晰,易理解.
8. 当你读取和写入数据时,使用适当的单位

2.14.2 MATLAB 总结

下面的总结列举了本章出现的所有特殊符号,命令和函数,后面跟的是简短的描述.

特殊符号

符号	说明
[ ]	数组构造器
( )	用来装载下标
' '	用来限制一个字符串
,	分开下标,或分开元素
;	1.防止在命令窗口的重复 2.分开矩阵的行 3.在一行内分开几个赋值语句
%	标志注释的开始
:	克隆运算符
+	数组和矩阵的加法
-	数组和矩阵的减法
.*	数组乘法
*	矩阵乘法
./	数组右除法
.\	数组左除法
/	矩阵右除法
\	矩阵左除法
.^	数组幂运算
'	转义运算符命令和函数
…	且来表示语句太长,转到第二行写
abs(x)	计算 x 的绝对值
acos(x)	计算 x 的反余弦函数
angle（x）	计算复数 x 的幅角
asin（x）	计算 x 的反正弦函数值
atan（x）	计算 x 的反正切函数值
atan2（y，x）	
cos（x）cosx	
exp（x）	

符号	说明
log（x）	
[value,index]=max（x）	返回 x 中的最大值，和他所处的位置
[value,index]=min（x）	返回 x 中的最小值，和他所处的位置
mod（x,y）余数，	
sin（x）sinx	
sqrt（x）	x 的平方根
tan（x）tanx	
rounding(取整)函数	
ceil(x)	
fix(x)	
round(x)	
字符转换函数	
char(x)	将矩阵中的数转化为字符,矩阵中的元素就不大于 127
double(x)	将子符串转化为矩阵
int2str(x)	将整数 x 转化为字符串形式
num2str(x)	将带小数点的数转化为一个字符型数组
str2num(x)	将字符串转化为数
format short	保留小数点后 4 位（默认格式）
format long	保留小数点后 14 位
format short e	带有 5 位有效数字科学记数法
format short g	总共有 5 个数字，可以用科学记数法，也可不用
format long e	带有 15 位有效数字科学记数法
format long g	总共有 5 个数字，可以用科学记数法，也可不用
format bank	美元格式
format hex	用 16 进制表示
format rat	两个小整数的比
format compact	隐藏多余的换行符
format loose	使用多余的换行符
format +	只显示这个数的正负
pi	有 15 个有效值的 π
i,j	代表虚数 i($\sqrt{-1}$)
Inf	这个符号代表无穷大，它一般情况下是除以 0 产生的
NaN	这个符号代表没有这个数。它一般由数学运算得到的。例如，0 除以 0。
clock	这个特殊变量包含了当前的年，月，日，时，分，秒，是一个 6 元素行向量
date	包含当前的日期，是用的字符形式
eps	变量名是 epsilon 的简写。它代表计算能机辨别的两数之间的最小数
ans	常用于存储表达式的结果，如果这个结果没有明确的赋值于某个变量
char	字符型
plot	函数的 x,y 均用线性尺度
semilog	函数 x 轴用对数尺度，y 轴将用线性尺度
Semiloge	函数 x 轴用线性尺度，y 轴用对数尺度
loglog	函数两坐标轴将会都用对数尺度。

2.15 练习

2.1 看下面的数组回答有关问题

$$array1=\begin{bmatrix}1.1 & 0.0 & 2.1 & -3.5 & 6.0\\ 0.0 & 1.1 & -6.6 & 2.8 & 3.4\\ 2.1 & 0.1 & 0.3 & -0.4 & 1.3\\ -1.4 & 5.1 & 0.0 & 1.1 & 0.0\end{bmatrix}$$

a. array1 的大小是多少?
b. array1(4,1)的值是多少?
c. array1(:,1:2)的大小和值为多少?
d. array1([1 3],end)的大小和值为多少?

2.2 下面的变量名那些合法那些不合法.为什么?

a. dog1 b. 1dog c. Do_you_know_the_way_to_san_jose
d. _help e. What's_up?

2.3 写出下面的数组的大小和内容.注意后面的数组可能根据前面数组的定义.

a. a=1:2:5;
b. b=[a' a' a'];
c. c=b(1:2:3,1:2:3);
d. d=a+b(2,:);
e. w=[zeros(1,3) ones(3,1)' 3:5'];
f. b([1 3],2)=b([3 1],2);

2.4 数组定义如下,写下面的子数组的内容

$$array1=\begin{bmatrix}1.1 & 0.0 & 2.1 & -3.5 & 6.0\\ 0.0 & 1.1 & -6.6 & 2.8 & 3.4\\ 2.1 & 0.1 & 0.3 & -0.4 & 1.3\\ -1.4 & 5.1 & 0.0 & 1.1 & 0.0\end{bmatrix}$$

a. array1(3,:)
b. array1(:,3)
c. array1(1:2:3,[3 3 4])
d. array1([1 1],:)

2.5 已知 value 的初始化值是 10π,写出下列语句的输出

```
disp(['value = ' num2str(value)]);
disp(['value = ' int2str(value)]);
fprintf('value = %e\n',value);
fprintf('value = %f\n',value);
fprintf('value = %g\n',value);
fprintf('value = %12.4f\n',value);
```

2.6 a,b,c 的定义如下,如果下面运算是合法的,那么写出结果,如果不合法,说出原因.

$$a=\begin{bmatrix}2 & -2\\ -1 & 2\end{bmatrix} \qquad b=\begin{bmatrix}1 & -1\\ 0 & 2\end{bmatrix}$$

$$a=\begin{bmatrix}1\\ -2\end{bmatrix} \qquad d=eye(2)$$

a. result=a+b; b. result=a*d; c. result=a.*d; d. result=a*c;
e. result=a.*c; f. result=a\b; g. result=a.\b; h. result=a .^ b;

2.7 求下列表达式的值

a. 11/5+6 b. (11/5)+b c. 11/(5+b)
d. 3^2^3 e. 3^(2^3) f. (3^2)^3
g. round(-11/5)+6 h. ceil(-11/5)+6 i. floor(-11/5)+6

2.8 用 **MATLAB** 计算下列表达式的值

a.(3-5i)(-4+6i)　　　　　　　b. $\cos^{-1}(1.2)$

2.9 求下列联立方程组中的各 x 的值

$$-2.0x_1+5.0x_2+1.0x_3+3.0x_4+4.0x_5-1.0x_6=0.0$$
$$2.0x_1-1.0x_2-5.0x_3-2.0x_4+6.0x_5+4.0x_6=1.0$$
$$-1.0x_1+6.0x_2-4.0x_3-5.0x_4+3.0x_5-1.0x_6=-6.0$$
$$4.0x_1+3.0x_2-6.0x_3-5.0x_4-2.0x_5-2.0x_6=10.0$$
$$-3.0x_1+6.0x_2+4.0x_3+2.0x_4-6.0x_5+4.0x_6=-6.0$$
$$2.0x_1+4.0x_2+4.0x_3+4.0x_4+5.0x_5-4.0x_6=-2.0$$

2.10 球的位置和速度.如果一静止小球在离地 h_0 的地方以初速度 v_0 做垂直运动,其等式为

$$h(t)=\frac{1}{2}gt^2+v_o t+h_o \qquad (2.10)$$

$$v(t)=gt+v_o \qquad (2.11)$$

其中 g 为重力加速度(- 9.81m/(s*s)),$h(t)$代表在 t 时刻小球的高度.$v(t)$代表在时刻 t 小球的速度.编写一个 **MATLAB** 程序,计算出每一秒钟的速度和高度,并打印出 h,v 关于时间 t 的函数.确保在你的图中有合适的标签.

2.11 编写一个程序,计算出坐标系中用户指定两点(X1,Y1)和(X2,Y2)之间的距离.在编个程序时,注意培养好的编程习惯.这两点之间距离的计算公式为

$$d=\sqrt{(x1-x2)^2+(y1-y2)^2} \qquad (2.12)$$

2.12 工程师们经常用分贝或 dB 来描述两功率之比.1dB 的定义如下

$$dB=10\log_{10}\frac{P_2}{P_1} \qquad (2.13)$$

P_2 是已测量的功率,P_1 代表参考功率.

a. 假设参考功率 P_1 为 1mw,编写一个程序,接受一个输入功率 P_2 并把转化成为以 1mw 为参考功率的 dB.(它在工程上有一个特殊单位 dBm).在编写程序时,注意培养好的编程习惯.

b. 写一个程序,创建一个以 W 为单位的功率的相对功率(单位为 dBm)的图象.第一个图象的 XY 轴都要用线性轴.而第二图象要用对数- 线性 xy 轴.

2.13 双曲余弦.

双曲余弦的定义如下

$$\cosh x=\frac{e^x+e^{-x}}{2} \qquad (2.14)$$

编写一个程序,计算出用户指定的 x 的值对应的双曲余弦值.用这个程序计算 3.0 的双曲余弦值.和 **MATLAB** 中的内建函数 cosh(x)得到的值是否完全相同.用 **MATLAB** 打印出这个函数的图象.当 x 为何值时,这个函数有最小值?最小值为多少?

2.14 弹簧中的能量.压缩弹簧弹力的大小可由下面的公式计算出来

$$F=kx \qquad (2.15)$$

F 代表弹力,单位为 N.k 代表劲度系数单位为 N/m.存储在压力弹簧中的势能为

$$E=\frac{1}{2}kx^2 \qquad (2.16)$$

E 代表势能,单位为焦.下面是 4 个可用压缩弹簧的信息.

	弹簧 1	弹簧 2	弹簧 3	弹簧 4
力/(N)	20	24	22	20
弹簧劲度系数 k(N/m)	500	600	700	800

编写一个程序,计算出每一个弹簧的压缩量和弹力势能.哪一个弹簧的弹力最大?

2.15 收音接收机.

一个最简单的调幅收音接收机如图2.13所示.这个接收机由一个*RLC*振荡电路组成,这个振荡电路包括一个电阻,一个电容和一个电感,还有连接它们的导线.就像图中所描绘的,将这个 *RLC* 电路连接到天线和大地.

这个振荡电路保证了这个接收机在 AM 波段中的众多的电台中接收到特定的一个台.只有在共振频率下,接收的信号最强.*LC* 电路的共振频率公式为

$$f_o = \frac{1}{2\pi\sqrt{LC}} \qquad (2.17)$$

L 代表电感,单们为 H,*C* 代表电容,单们为 F.编写一个程序,输入 *L* 和 *C* 的值,计算出它的共振频率.用 *L*=0.1mH,*C*=0.5nF 来检测这个程序,计算共振频率.

2.16 收音接收机.电阻上的电压可通过频率计算出来,公式如下

$$V_R = \frac{R}{\sqrt{R^2 + (\omega L - \dfrac{1}{\omega C})^2}} V_o \qquad (2.18)$$

$\omega=2\pi f$,以 Hz 为单位的频率.假设 L=0.1mH,C=0.25nF,R=50Ω,V_0=10mV.

a. 画出以频率为自变量的电阻电压函数.在什么频率下,电阻上的电压最大?这时的电压为多少?这个频率叫做电路的固有频率.

b. 如果这个频率比固有频率大百分之十,此时电阻上的电压为多少?这个收音接收机是如何选台的?

c. 在什么频率下这个电阻上的电压会降到固有频率电压的一半?

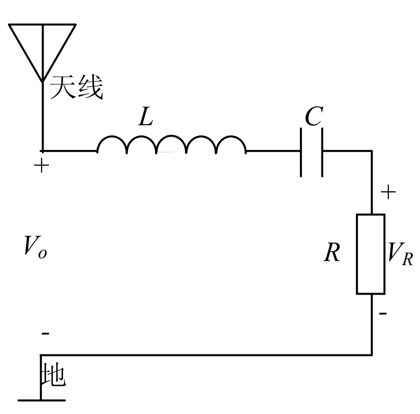


图 2.13 简易接收机原理图

2.17 假设两个信号同时被天线接收.其中一个信号的大小为 1V,频率为 1000kHz,而另一个信号的大小为 1V,950kHz.第一个信号给负载 *R* 的功率是多少?第二个信号给负载*R*的功率是多少?计算第二个信号相对第二个信号的增益或衰减.与第一个信号相比,第二个信号增益或衰减了多少?

2.18 飞船的运转半径.图 2.14 向大家显示的是一个做匀速圆周运动的物体.

其向心加速度公式为

$$a = \frac{v^2}{r} \qquad (2.19)$$

a 代表向心加速度,单位为 m/s^2.*v* 代表物体运动的速率,单位为 m/s,*r* 代表半径,单位为 m.假设这个物体是一个飞机,回答下列问题:

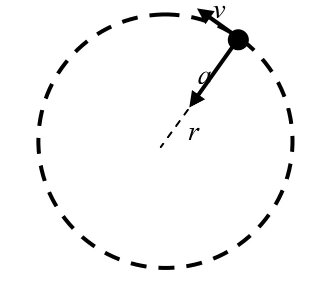


图 2.4 做匀速圆周运动的物体

a. 假设飞机的运动速度为 0.85 马赫,即声速的85%.如果向心加速度为 2*g*,那么飞机的半径为多少?

b. 假设飞行速度增大到 1.5 马赫,那么飞机的半径为多少?

c. 运转半径是飞机飞行速度的一个函数,自变量的定义域为 0.5 马赫到 2.0 马赫,向心加速度仍为 2*g*,画出这个函数的图象.

d. 假设飞行员能忍耐的最大加速度为 7*g*.那么以 1.5 马赫飞行的最小半径为多少?

e. 画出以向心加速度为自变量的半径函数,向心加速度的取值为[2*g*,6*g*],假设运转速度为 0.85 马赫.

第三章 分支语句和编程设计

在前面的章节中，我们开发了几个完全运转的 **MATLAB** 程序。但是这些程序都十分简单，包括一系列的 **MATLAB** 语句，这些语句按照固定的顺序一个接一个的执行。像这样的程序我们称之顺序结构程序。它首先读取输入，然后运算得到所需结果，打印出结果，并退出。至于要多次重复运算程序的某些部分是没有办法的，也不能根据输入的值，有选择地执行程序的某些部分。

在下面的两章中，我们将向大家介绍大量的 **MATLAB** 语句，这些语句允许我们来控制中语句的执行顺序。有两大类控制顺序结构：**选择结构**，用选择执行特定的语句；**循环结构**，用于重复执行特定部分的代码。选择结构将会本章讨论，循环结构我们将会在第四章讨论。

随着选择和循环介绍，我们的程序也将变得复杂，对于解决问题来说，将会变得简单。为了帮助大家避免在编程过程中出现大量的错误，我们将向大家介绍正式的编程步骤，即自上而下的编程方法。我们也会向大家介绍一些普通的算法开发工具即伪代码。

3.1 自上而下的编程方法简介

假设你是在工厂工作的工程师，为了解决某些问题，你要编写一个程序。你如何开始呢？当遇到一个新问题时，我们的心里会自然而然的产生这样的想法：马上坐在计算机前，开始编程，而不用浪费大量的时间思考我们所要解决的问题是什么？用这种不切实际的想法来编一些非常小的程序可能会成功。但在现实中，问题可能会非常的大，程序员再用这种方法编程将会陷入困境。对于一个大的程序来说，在编写代码之前你要通盘的思考你所要面临的问题和解决的方法。在本节中，我们将向大家介绍正式的编程设计步骤，然后应用这个步骤来编写本书所有的大的应用程序。对于我们所遇到一些简单的例子来说，这个步骤好像有些画蛇添足。但是当我们解决的问题变得越来越大的时侯，这个步骤将会变得异常重要。

当我还没有毕业的时侯，一个教授喜欢说："编程很简单，因为我知道在编程的过程的困难"。当我们离开学校，在工厂从事于大规模软件工程编写时，我深深地理解了它所说的话。我发现在工作中我遇到的大多数困难都是对所要解决问题的理解。一旦你真正理解了问题，你就会把这个问题分解成许多小的问题，更加易于管理的小块，然后逐一解决某一个小块。自上而下的编程方法是我们正规编程设计的基础。我们现在向大家介绍这些在图 3.1 说明的步骤细节。步骤如下:

1.清晰地陈述你所要解决的问题

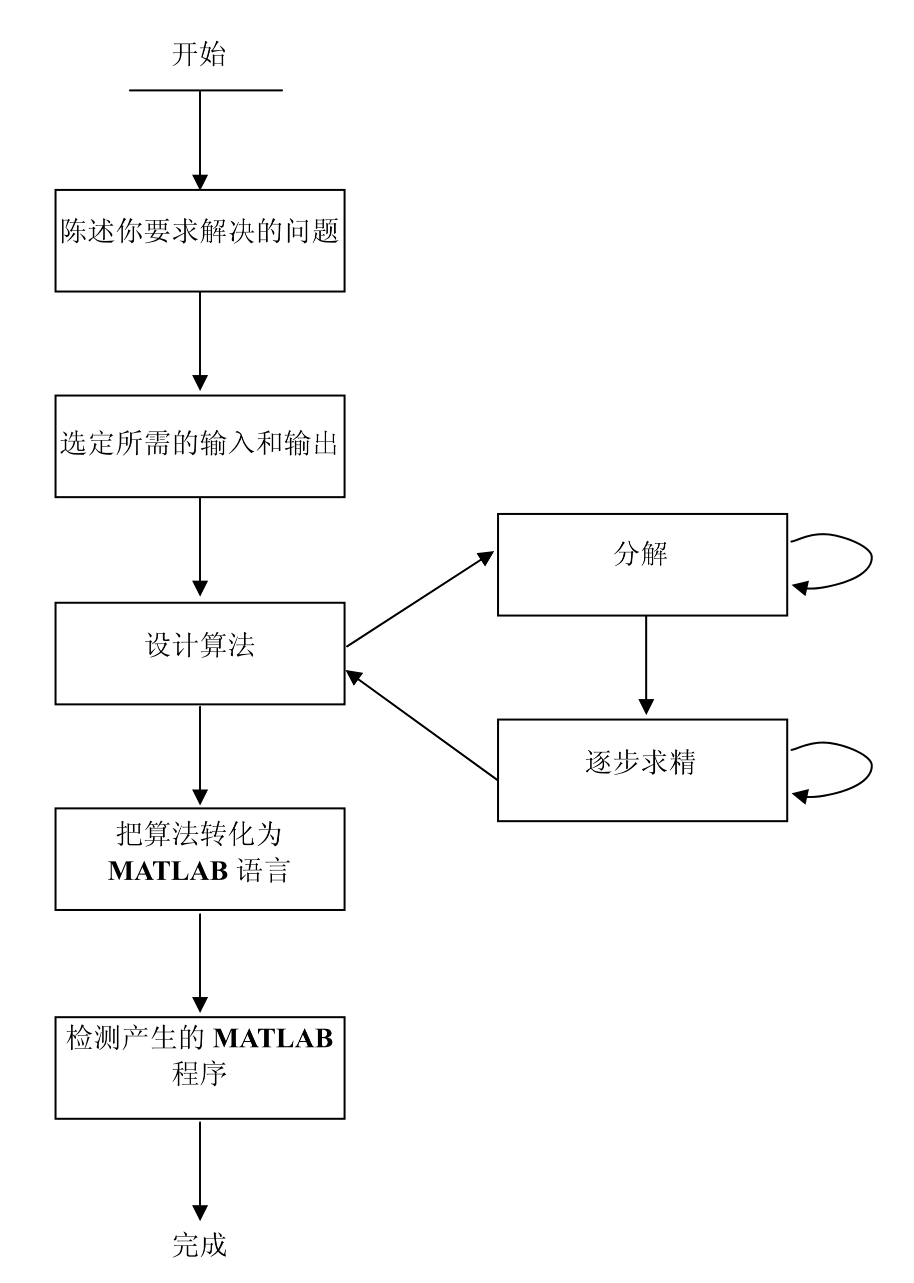
编写的程序大多数情况下要满足一些感觉上的需要，但这种需要不一定能够被人清晰地表达出来。例如，用户需要一个解线性方程组的表达式。像这样的要求就不够清楚，程序员就很难编出一个使他满意的程序。他必须弄清楚要有多少问题需要解决?在这些方程式中有没有对称的形式使我们的开发变得简单?程序设计者必须和使用者讨论所需的程序，他们必须要对完成的任务有一个精确细致的描述。对问题清晰的描述可以防止误解，并且能够帮助程序员合理的组织他的思想。上面的例子对问题合适的陈述应为:

设计一个用于解决联立线性方程组的程序，这些方程中未知数的系数为实数，最多有 20 个未知数。

2.定义程序所需的输入量和程序所产生的输出量

指定输入量和输出量，只有这样新的程序才能适应全过程计划。在这个例子中方程式的系数可能有其预先存在的顺序，我们的新程序必须能按照顺序读取它们。相似地，也需要产生出这个程序所要求的结果，即输出量，我们还要以一定的格式打印出来。

3.设计你的程序得以实现的算法



算法是指为某个问题找到答案一步接一步的程序。在这个阶段自上而下的编程方法发挥了作用。编程设计者开始对这个问题进行逻辑划分，把它逐步分解为一个又一个子工作。这个过程叫做分解(decomposition)。如果一些子工作还是比较大，设计者还可以把他它分解成更小的块。这个过程将会继续到问题被分解成许多简单且易理解的小块为止。

在问题被分解成小块之后，每一个小块要被进一步的求精，这个过程叫做逐步求精(stepwise refinement)。在这个过程中，设计者开始于对本小块代码总括性的描述，然后开始一步一步地定义所需的函数，越来越具体，直到他能够转化为 **MATLAB** 语句。逐步求精的过程中，我们要用到的伪代码将会在下节为大家介绍。

在算法开发过程中，这个方法是非常有用的。如果设计者真正理解了解决问题这个些步骤，他将会对问题进行分解和逐步求精。

4.把算法转化为代码

如果分解和逐步求精的过程已经顺利完成，那么这一步将会异常地简单。所有程序员都会将伪代码一句一句地转化为合适地 **MATLAB** 语句。

5 检测产生的 **MATLAB** 程序

这一步是真正的拦路虎。首先，程序的每一部分将会被单独地检测，如果有可能的话，整个程序还要被检测一遍。在我们检测程序时，我们必须证明所有合法输入数据值都能够正

常运行。用标准的输入值检测程序，看它是否产生了值。如果在一个程序中执行的算法包含了不同的分支，你必须检测每一个分支，以保证产生正确的答案。大程序在交付大众使用之前，必须经过一系列地检测(图 3.2)。检测的第一步有时被称为单元检测(unit testing)。在单元检测过程中，程序的子程序将会被独立地检测以证明它的正确性。当单元检测结束之后，这个程序将进行一系列的组合，把独立的子程序联合产生出最后的程序。程序第一步的联合通常只包括很少的子程序。通过组合这些子程序，经常用检查子程序或函数之间的联系。在一系列地组合过程中，越来越多的子程序被加了进来，直到整个程序的完成。在每一次组合的过程中，每一个错误都会被发现并在进行下一次组合之前纠正过来。

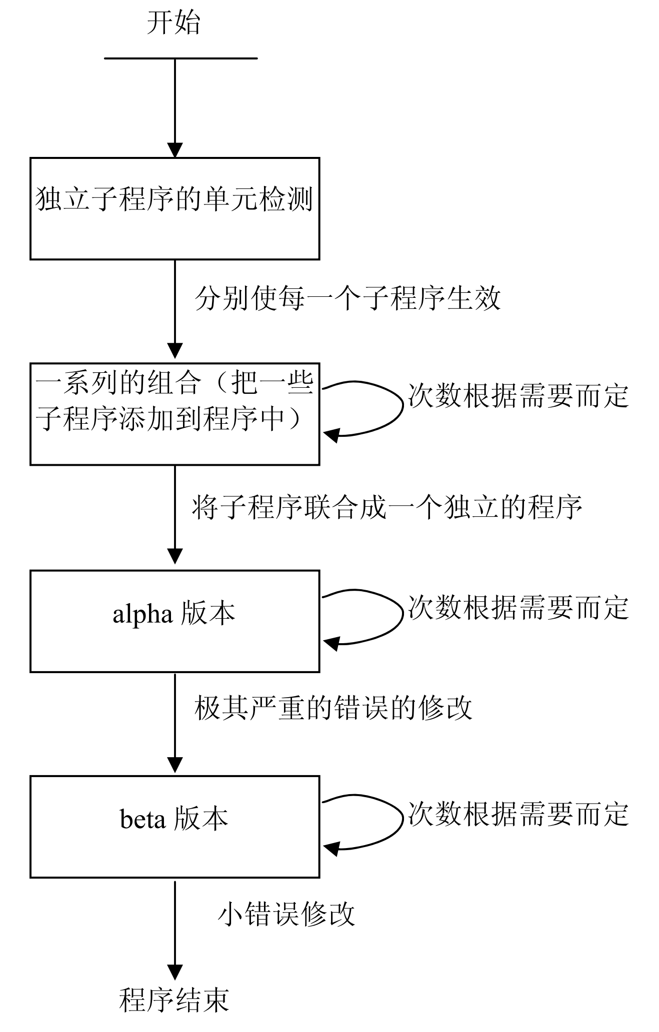


图 3.2 大程序典型地调试过程

在整个程序被组合之后，调试继续进行。程序第一个版本我们通常称之为“alpha 版本”。程序员和其他有机会接近它的人可以想尽一切办法应用它，以发现其中的漏洞，然后改正之。当许许多多大的错误从程序中去除，一个新的版本出现了，我们称之“beta 版本”。beta 版本就要公开地发行给天天需要这个程序工作的人。这些用户使这个程序在不同的环境下，在不同的输入条件下工作，会发现许多的错误，并报告给程序员。当这些错误被更正后，这个程序就能够发行给公众使用了。因为本书中的程序都比较小，没有必要进行上述的大规模的检测。但是我们会遵循基本的调试原则。

程序设计的基本步骤如下:

1.清晰地陈述出你要解决的问题。
2.确定程序所需地输入量和程序所产生的输出量。
3.为你的程序设计算法
4.将算法转化为 **MATLAB** 语句
5.调试 **MATLAB** 程序

好的编程习惯
遵循上面的步骤编写可靠，易理解的 MATLAB 程序。

在大的编程项目中，花在编程序的时间是出奇的少。Frederick P Brooks 在他的 the Mythical Man-Month 书中写道，对于大的软件工程来说，三分之一的时间花在计划如何来做上(第一步到第三步)，六分之一的时间花在编写程序上，近一半的时间用来调试程序。而我们能做的只有压缩调试用的时间。在计划阶段做好充分的准备和在编程过程使用良好的编程习惯，这样会大大降低我们调试所用的时间。好的编程习惯能减少出错的数量，也能使别人迅速地找出其中的错误。

3.2 伪代码的应用

作为我们设计步骤的一部分，描述出你要执行的算法是非常必要的。算法的描述有一种标准形式，能让你和大家都能理解，这种描述将帮助你的内容转化为 **MATLAB** 代码。我们用于描述算法的标准形式叫做构造(constructs 有时也称 structure)。用这些结构描述出的算法，我们称之为结构化算法。当在我们在 **MATLAB** 程序中执行这个算法时，产生的程序叫做结构化程序。

我们可以用伪代码的形式建立算法的结构。伪代码是 **MATLAB** 和英语的混合体。和 **MATLAB** 一样，它是结构化的，一行表达一个明确的意思或代码的片段，但每一行的描述用的是英语或其他人类语言。伪代码的每一行都应用普通简单且易于理解的英语或中文描述。因为修改简单灵活，所以伪代码在开发算法的过程中非常的有用。因为伪代码给编辑器或字处理器(通常用于编写 **MATLAB** 程序)的，而不需要其他的可视化功能。例如下面是例 2.3 的算法伪代码

```
Prompt user to enter temperature in degrees Fahrenheit
Read temperature in degrees Fahrenheit(temp_f)
temp_k in kelvins ← (5/9) * (temp_f - 32) + 273.15
Write temperature in kelvins
```

注意用向左指的箭头←替代等号(=)指出一个值将存储到对应的变量中，这样就避免了赋值号与等号的混淆。在把它们转化为 **MATLAB** 代码之前，伪代码将有助于你思想的组织。

3.3 关系运算符和逻辑运算符

选择结构的运算由一个表达式控制的，这个表达式的结果只有 true(1)和 false(0)。有两种形式的运算符可以在 **MATLAB** 中关系得到 true/false：关系运算符和逻辑运算符。

跟 C 语言一样，**MATLAB** 没有布尔型和逻辑数据类型。**MATLAB** 把 0 值作为结果 false，把所有的非 0 值作为结果 ture。

3.3.1 关系运算符

关系运算符是指两数值或字符操作数的运算符，这种运算将会根椐两操作数的关系产生结果 true 或 false。关系运算的基本形式如下

$$a_1 \ op \ a_2$$

其中 a_1 和 a_2 是算术表达式，变量或字符串，op 代表表 3.1 中的关系运算符中的一个。如果两者的关系为真（true）时，那么这个运算将会返回 1 值；否则将会返回 0 值。

表 3.1 关系运算符

运算符	运算
==	等于
~=	不等于
>	大于
>=	大于或等于
<	小于
<=	小于或等于

下面是一些关系运算和它的结果运算结果

运算	结果
3 < 4	1
3 <= 4	1
3 == 4	0
3 > 4	0
4 <= 4	1
'A' < 'B'	1

最后一个运算得到的结果为 1，是因为字符之间的求值要按照子母表的顺序。

关系运算符也可用于标量与数组的比较。例如，如果 $a=\begin{bmatrix}1 & 0\\-2 & 1\end{bmatrix}$ 和 b=0，那么表达式 a>b 将会产生结果 $\begin{bmatrix}1 & 0\\1 & 0\end{bmatrix}$。关系运算符也可比较两个关系运算符，只要两个数组具有相同的大小。例如 $a=\begin{bmatrix}1 & 0\\-2 & 1\end{bmatrix}$，$b=\begin{bmatrix}0 & 2\\-2 & -1\end{bmatrix}$，表达式 a>=b 将会产生结果 $\begin{bmatrix}1 & 0\\1 & 1\end{bmatrix}$。如果这个数组具有不同的大小，那么将会产生运行时错误。

注意因为字符串实际上是字符的数组，关系运算符也比较两个相同长度的字符串。如果它们有不同的长度，比较运算将会产生一个错误。在第六章中我们将会学到一个更普遍的方法。等于关系运算符由两个等号组成，而赋值运算符只有一个等号。它们是完全不同的两个符号，初学者极易混淆。符号==是一个比较运算符，返回一个逻辑数，而符号=是将等号右边的表达式的值赋给左边的变量。当进行比较运算的时候，初学者经常用误用符号=。

编程隐患
小心谨慎不要混淆了等于关系运算符（==）和赋值运算符（=）。

在运算的层次中，关系运算在所有数学运算的之后进行。所以下面两个表达式是等价的，均产生结果 1。

7 + 3 < 2 + 11
(7 + 3)< (2 + 11)

3.3.2 小心==和~=运算符

等于运算符（==）如果两变量值相同将会返回变量值 1，如果不同将返回 0。
不等运算符（~=）如果两变量值不同则返回 1，相则返回 0。

用这两个运算符比较两个字符串他是安全的，不会出现错误。但对两个数字数据的比较，将可能产生异想不到的错误。两个理论上相等的数不能有一丝一毫的差别，而在计算机计算的过程中出现了近似的现象，从而可能在判断相等与不相等的过程中产生错误，这种错误叫做 round off 错误。例如，考虑下面的两个数，两者均应等于 0。

```
a = 0;
b = sin(pi);
```

因为这两个数在理论上相等的，所以关系式 a==b 应当返回值 1。但在事实上，**MATLAB**

计算所产生的结果的是

```
>> a = 0;
>> b = sin(pi);
>> a == b
ans =
     0
```

MATLAB 报告了 a 和 b 不同因为他产生了一个 round off 错误，在计算中 sin(pi)产生了结果 1.2246×10^{-16} 而不是 0。两个理论上相等的值因为 round off 错误而失之发生了细微的差别。

我们可以通过检测两数之间在一定的范围内是不是近似相等，在这个精确范围内可能会产生 round off 错误。例如测试

```
>> abs(a - b) < 1.0E-14
ans =
     1
```

将会产生正确的结果，不管在 a 与 b 的计算中产不产生的 round off 错误。

好的编程习惯

在我们检测两数值是否相等时一定要小心，因为 round off 错误可能会使两个本来应该相等的值不相等了。这时你可以在 round off 错误的范围内它是不是近似相等。

3.3.3 逻辑运算符

逻辑运算符是联系一个或二个逻辑操作数并能产生一个逻辑结果的运算符。有三个二元运算符：分别为 AND，OR 和异或运算符，还有一个一元运算符 NOT。二元逻辑运算的基本形式

$$l_1 \text{ op } l_2$$

一元逻辑运算的基本形式为

$$\text{op } l_1$$

l_1 和 l_2 代表表达式或变量，op 代表表 3.2 中的逻辑运算符。如果 l_1 和 l_2 的逻辑运算关系为 true，那么运算将会返回值 1，否则将会产生 0。

表 3.2 逻辑运算符

&	逻辑与
|	逻辑或
xor	逻辑与或
~	逻辑非

运算的结果总结在真值表 3.3 中，它向我们展示每一种运算所有可能的结果。如果一个数的值不为 0，那么 **MATLAB** 将把看作 true，如

表 3.3 逻辑真值表

输入		与	或	异或	非
l_1	l_2	$l_1 \& l_2$	$l_1 \mid l_2$	$xor(l_1,l_2)$	$\sim l_1$
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

果它为 0，则其为 false。所以~5 的结果为 0，~0 的结果为 1。

标量和数组之间也可进行逻辑运算。例如，$a=\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$，b=0，那么表达式 a&b 将会产生结果$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$。两数组之间也可进行逻辑运算，只要它们具有相同的大小。例如，$a=\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$，$b=\begin{bmatrix}1 & 1\\0 & 0\end{bmatrix}$则 a|b 产生的结果$\begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}$。如果两个数的大小不相同，那么将会产生运行时错误。

在运算的顺序中，逻辑运算在所有的数学运算和关系运算之后进行。

表达式中的运算顺序如下：
1.所有的数学运算按照前面描述的顺序的进行。
2.从左向右依次进行关系运算
3.执行所有~运算
4.从左向右依次进行&运算
5.从左向右依次进行|运算和数学运算一样，括号能够改变括号的默认顺序。

下面是关于逻辑运算的一些例子。

例 3.1

假设下面有三个变量被初始和一些表达式及其运算结果。
value1 = 1
value2 = 0
value3 = -10

逻辑表达式	结果
(a) ~value1	0
(b) value1 | value2	1
(c) value1 & value2	0
(d) value1 & value2 | value3	1
(e) value1 & (value2 | value3)	1
(f) ~(value1 & value3)	0

因为~运算在其它的逻辑运算之前进行，那么(f)中的括号是必须的。如果去掉括号的话，(f)表达式将等价于(~value1)&value3。

3.3.4 逻辑函数

MATLAB 中有大量的逻辑函数，在条件满足时，函数返回 1。条件不满足时，返回 0。这些函数和逻辑运算与关系联合在组成选择结构和循环结构。表 3.4 列出了一系列的逻辑函数。

表 3.4 **MATLAB** 逻辑函数

函数	用途
ischar(a)	a 是字符数组返回 1，否则返回 0
isempty(a)	a 是空数组返回 1，否则返回 0
isinf(a)	a 是无穷大，则返回 1，否则返回 0
isnan(a)	a 不是一个数则返 1，否则返回 0
isnumeric(a)	a 是一个数值数组返回 1，否则返回 0

测试 3.1

本测试提供了一个快速的检查方式，看你是否掌握了 3.3 的基本内容。如果你对本测试有疑问，你可以重读 3.3，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

变量 a，b，c，d 定义如下，计算后面的表达式。

a = 20;　　b = -2;
c = 0;　　d = 1;

1. a > b　　2. b > d;
3. a > b & c > d　　4. a == b
5. a & b > c　　6. ~~b

变量 a，b，c，d 定义如下，计算后面的表达式。

a = 2;　　$b = \begin{bmatrix} 1 & -2 \\ -0 & 10 \end{bmatrix}$

$c = \begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix}$　　$d = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 1 & 0 \end{bmatrix}$

7. ~(a > b);
8. a > c & b > c;
9. c <= d

变量 a，b，c，d 定义如下，计算后面的表达式。

a = 2;　　b = 3;
c = 10;　　d = 0;

10. a*b^2 > a*c
11. d | b b > a
12. (d | b) > a

变量 a，b，c，d 定义如下，计算后面的表达式。

a = 20;　　b = -2;
c = 0;　　d = 'Test';

13. isinf(a/b)
14. isinf(a/c)
15. a > b & ischar(d)
16. isempty(c)

3.4 选择结构(分支语句)

选择结构可以使 **MATLAB** 选择性执行指定区域内的代码(称之为语句块 blocks)，而跳过其他区域的代码。选择结构在 **MATLAB** 中有三种具体的形式:if 结构，switch 结构和 try/catch 结构。

3.4.1 if 结构

if 结构的基本形式如下:

```
if control_expr_1
    Statement 1 ┐
    Statement 2 ├ Block 1
    ...         ┘
elseif control_expr_2
```

```
    Statement 1 ⎫
    Statement 2 ⎬ Block 2
    ...         ⎭
else
    Statement 1 ⎫
    Statement 2 ⎬ Block 3
    ...         ⎭
end
```

其中 control expression 控制 if 结构的运算。如果 control_expr_1 的值非 0，那么程序将会执行语句块 1(block1)，然后跳到 end 后面的第一个可执行语句继续执行。否则，程序将会检测 control_expr_2 的值，。如果 control_expr_2 的值非 0，那么程序将会执行语句块 2(block2)，然后跳到 end 后面的第一个可执行语句继续执行。如果所有的控制表达式(control expression)均为 0，那么程序将会执行与 else 相关的语句块。

在一个 if 结构中，可以有任意个 elseif 语句，但 else 语句最多有一个。只要上面每一个控制表达式均为 0，那么下一个控制表达式将会被检测。一旦其中的一个表达式的值非 0，对应的语句块就要被执行，然后跳到 end 后面的第一个可执行语句继续执行。如果所有的控制表达式(controlexpression)均为 0，那么程序将会执行 else 语句。如果没有 else 语句，程序将会执行 end 后面的语句，而不执行 if 结构中的部分。

注意 **MATLAB** 在 if 结构中的关键字 end 与第二章中提到的返回已知下标最大值函数 end 完全不同。**MATLAB** 通过 end 在 M 文件中的上下文来区分开它的两个用途。在大多数情况下，控制表达式均可以联合关系运算符和逻辑运算符。正像我们在本章早些时侯学到的，当对应的条件为真时，关系运算和逻辑运算将会产生 1，否则产生 0。所以当一个运算条件为真时，运算结果为非 0，则对应的语句块，就会被执行。

例如，一元二次方程的基本形式如下:

$$ax^2 + bx + c = 0 \tag{3.1}$$

其解为

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.2}$$

其中 $b^2 - 4ac$ 是我们熟知的判别式，当 $b^2 - 4ac>0$ 时，方程式有两个不同的实数根，当 $b^2 - 4ac=0$，有两个相同的实数根，当 $b^2 - 4ac<0$ 时，方程式有两个不同的复根。

假设我们检测某一元二次根的情况，并告诉使用者这个方程有两个复根，还是两个相等的实根和两个不相等的实根。用伪代码这个结构的形式如下:

```
if (b^2 - 4*a*c) < 0
    Write msg that equation has two complex roots.
elseif (b^2 - 4*a*c) ==0
    Write msg that equation has two identical real roots.
else
    Write msg that equation has two distinct real roots.
end
```

转化为 **MATLAB** 语言:

```
if (b^2 - 4*a*c) < 0
    disp('This equation has two complex roots.');
elseif (b^2 - 4*a*c) == 0
    disp('This equation has two identical real roots.');
else
    disp('This equation has two distinct real roots.');
end
```

回忆一下，判断为真时，关系运算符将会返回一个非 0 值，从而导致对应语句的执行。

为增加程序的可读性，在 if 结构中的语句块中最好缩进 2 到 3 个空格，而实际上没有必要。

好的编程习惯

if 结构体经常缩进 2 到 3 个空格，以增强程序的可读性。

你可以在一行内写完一个完整的 if 结构，只需把结构的每一部分后面加上分号或逗号，所以下面的两个结构是等价的:

```
if x < 0
    y = abs(x);
end
```

和

```
if x < 0; y = abs(x); end
```

但是这种方式只适用于简单的结构。

3.4.2 if 结构举例

我们将用两个例子来说明 if 结构的用途

例 3.2

求一元二次方程的根

设计并编写一个程序，用来求解一元二次方程的根。

答案:

我们将本章开头介绍的方法进行编程。

1.陈述问题这个问题的陈述非常的简单，我们要求一元二次方程的根，不管它的根是实根还是复根，有一个根还是两个根。

2.定义输入和输出

本程序的输入应为系数 a，b，c

$$ax^2 + bx + c = 0 \tag{3.1}$$

输出量应为两个不相等的实数。两个相等的实数或两个复数。

3.写出算法本程序可分为三大块，它的函数分别为输入，运算过程和输出。

我们把每一个大块分解成更小的，更细微的工作。根据判别式的值，可能有三种计算途径，

```
读取输入的数据
计算出根
输入出根
```

所以我们要用到有三种选项的 if 结构。产生的伪代码如下

```
Prompt the user for the coefficients a, b, and c.
Read a, b, and c
discriminant ← b^2 - 4*a*c
if discriminat > 0
    x1 ← (-b + sqrt(discriminant)) / (2*a)
    x1 ← (-b - sqrt(discriminant)) / (2*a)
    Write msg that equation has two distinct real roots.
    Write out the two roots.
elseif discriminant == 0
    x1 ← -b / (2*a)
    Write msg that equation has two identical real roots.
    Write out the repeated roots.
else
    real_part ← -b / (2*a)
    imag_part ← sqrt(abs(discriminant)) / (2*a)
```

```
    Write msg that equation has two complex roots.
    Write out the two roots.
end
```

4.把算法转化为 **MATLAB** 语言

```
%Script file: calc_roots.m
%
% Purpose:
% This program solves for the roots of a quadratic equation
% of the form a*x^2 + b*x + c = 0. It calculates the answers
% regardless of the type of roots that the equation possesses.
%
% Record of revisions:
%       Date          Programmer          Description of change
%       ======        ===========         ================
%       12/04/98      S. J. Chapman       Original code
%
% Define variables:
% a                 --Coefficient of x^2 term of equation
% b             --Coefficient of x term of equation
% c             --Constant term of equation
% discriminant          --Discriminant of the equation
% imag_part         --Imag part of equation (for complex roots)
% real_part         --Real part of equation (for complex roots)
% x1                --First solution of equation (for real roots)
% x2                --Second solution of equation (for real roots)
% Prompt the user for the coefficients of the equation
disp ('This program solves for the roots of a quadratic ');
disp ('equation of the form A*X^2 + B*X + C = 0.');
a = input('Enter the coefficient A: ');
b = input('Enter the coefficient B: ');
c = input('Enter the coefficient C: ');
% Calculate discriminant
discriminant = b^2 - 4 * a * c;
% Solve for the roots, depending on the vlaue of the discriminant.
if discriminant > 0    % there are two real roots, so ...
    x1 = (-b + sqrt(discriminant)) / (2*a);
    x2 = (-b - sqrt(discriminant)) / (2*a);
    disp('This equation has two real roots:');
    fprintf('x1 = %f\n', x1);
    fprintf('x2 = %f\n', x2);
elseif discriminant == 0 % there is one repeated root, so ...
    x1 = ( -b ) / (2*a);
    disp('This equation has two identical real roots:');
    fprintf('x1 = x2 = %f\n', x1);
else % there are complex roots, so ...
    real_part = (-b) / (2*a);
    imag_part = sqrt( abs(discriminant)) / (2*a);
    disp('This equation has complex roots:');
    fprintf('x1 = %f + i %f \n',real_part, imag_part);
    fprintf('x1 + %f - i %f \n', real_part, imag_part);
end
```

5.检测这个程序

下一步，我们必须输入实数来检测这个程序。因这个程序有三个可能的路径。所以在我们确信每一人路径都工作正常之前，必须把这三个路径检测一遍。从式子(3.2)中，我们可以有用下面的方法来验证程序的正确性。

x2 + 5x + 6 = 0　　　　　　x = -2, and x = -=3

x2 + 4x + 4 = 0 x = -2
x2 + 2x + 5 = 0 x = -1 ± i2

如果输入上面三个方程的系数得到对应的结果，则说明程序是正确的。

```
>> calc_roots
This program solves for the roots of a quadratic
equation of the form A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 5
Enter the coefficient C: 6
This equation has two real roots:
x1 = -2.000000
x2 = -3.000000
>> calc_roots
This program solves for the roots of a quadratic
equation of the form A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 4
Enter the coefficient C: 4
This equation has two identical real roots:
x1 = x2 = -2.000000
>> calc_roots
This program solves for the roots of a quadratic
equation of the form A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 2
Enter the coefficient C: 5
This equation has complex roots:
x1 = -1.000000 + i 2.000000
x1 + -1.000000 - i 2.000000
```

在三种不同的情况下，程序都给出了正确的结果。

例 3.3

编写一个程序，求以 x，y 为自变量函数 f(x，y)的值。函数 f(x，y)的定义如下:

$$f(x,y) = \begin{cases} x + y & x\geq 0 \text{ and } y\geq 0 \\ x + y^2 & x\geq 0 \text{ and } y<0 \\ x^2 + y & x<0 \text{ and } y\geq 0 \\ x^2 + y^2 & x<0 \text{ and } y<0 \end{cases}$$

答案:

根据自变量 x 和 y 的正负符号的不同，而采取不同的求值表达式。为选取合适的表达式，检查用户输入的 x，y 的正负符号是必要的。

1.陈述问题

这个问题的陈述非常简单:根据用户输入的 x，y，求函数 $f(x，y)$的值。

2.确定输入输出量

程序的输入量为函数的自变量 x，y。输出量为函数值 $f(x，y)$。

3.设计算法

这个问题可以把他分解成三个大块，即输入，计算过程，和输出。

我们把这三大块再分解成小的，精细的工作。在计算 f(x，y)时，我们有 4 种选择，选哪一种取决于 x，y 的值。所以及逻辑上我们要用 4 个选择的 if 结构来实现。产生的伪代码如下:

```
Prompt the user for the values x and y
```

```
Read x and y
if x≥0 and y≥0
    fun ← x + y
elseif x≥0 and y<0
    fun ← x + y^2
elseif x<0 and y≥0
    fun ← x^2 + y
else
    fun ← x^2 + y^2
end
Write out f(x,y)
```

4.转化为 **MATLAB** 语句

最终的代码如下

```
% Scripte file: funxy.m
%
% Purpose:
% This program solves the function f(x,y) for a
% user-specified x and y, where f(x,y) is defined as:
%                 _
%                 |
%                 | x + y                  x>=0 and y>=0
%                 | x + y^2            x>=0 and y<0
%         f(x,y)=     | x^2 + y              x<0 and y>=0
%                 | x^2 + y^2              x<0 and y<0
%                 |_
%
% Record of revisions:
%       Date          Programmer          Description of change
%       =====         =========           ================
%    12/05/98         S.J.Chapman         Original code
%
% Define variables:
% x              --First independent variable
% y              --Second independent variable
% fun                 --Resulting function
% Prompt the user for the values x and y
x = input('Enter the x coefficient: ');
y = input('Enter the y coefficient: ');
% Calculate the function f(x,y) based upon
% the signs of x and y.
if x>=0 & y>=0
    fun = x + y;
elseif x>=0 & y<0
    fun = x + y^2;
elseif x<0 & y>=0
    fun = x^2 + y;
else
    fun = x^2 + y^2;
end
% Write the value of the function.
disp(['The vlaue of the function is ' num2str(fun)]);
```

5.检测程序

下一步，我们必须输入实数来检测这个程序。因这个程序有四个可能的路径。所以在我们确信每一人路径都工作正常之前，必须把这四个路径检测一遍。我们分别取 4 个象限内的值(2，3)，(-2,3)，(2，-3)和(-2，-3)。我们用手工计算可得

$f(2,3) = 2 + 3 = 5$

$f(2,-3) = 2 + (-3)^2 = 11$
$f(-2,3) = (-2)^2 + 3 = 7$
$f(-2, -3) = (-2)^2 + (-3)^2 = 13$

当程序被编程后，运行 4 次并输入相应的值，运算结果如下:

```
>> funxy
Enter the x coefficient: 2
Enter the y coefficient: 3
The vlaue of the function is 5
>> funxy
Enter the x coefficient: 2
Enter the y coefficient: -3
The vlaue of the function is 11
>> funxy
Enter the x coefficient: -2
Enter the y coefficient: 3
The vlaue of the function is 7
>> funxy
Enter the x coefficient: -2
Enter the y coefficient: -3
The vlaue of the function is 13
```

程序在 4 种可能的情况下均产生了正确的结果。

3.4.3 关于 if 结构使用的注意事项

if 结构是非常灵活的，它必须含有一个 if 语句和一个 end 语句。中间可以有任意个 elseif 语句，也可以有一个 else 语句。联合它的这些特性，我们可以创建出我们需要的各种各样的选择结构。

还有 if 语句是可以嵌套的。如果 if 结构完全是另一个 if 结构的一个语句块，我们就称两者为嵌套关系。下面是两个 if 语句的嵌套。

```
if x > 0
    ...
    if y < 0
        ...
    end
    ...
end
```

MATLAB 翻译器经常把把已知的 end 语句和它最近的 if 语句联合在一起，所以第一个 end 语句和 if y<0 最靠近，而第二个 end 与 if x>0 最接近。对于一个编写正确的程序，它能工作正常。但如果程序员编写出错误，它将会使编译器出现混淆性错误信息提示。例如，假设我们编写一个大的程序，包括如下的一个结构:

```
...
if (test1)
    ...
    if (test2)
        ...
        if (test3)
            ...
        end
        ...
    end
    ...
end
```

这个程序包括了三个嵌套的 if 结构，在这个结构中可能有上千行的代码。现在假设第

一个 end 在编辑区域突然被删除，那么 **MATLAB** 编译器将会自动将第二个 end 与最里面的 if (test3)结构联合起来，第三个 end 将会和中间的 if(test2)联合起来。当编译器翻译到达文件结束的时候，那将发现第一个 if(test1)结构将永远没有结束，然后编译器就会产生一个错误提示信息，即缺少一个 end。但是，它不能告诉你问题发生在什么地方，这就使我们必须回过头去看整个程序，来找问题。

在大多数情况下，执行一个算法，即可以用多个 else if 语句，也可以用 if 语句的嵌套。在这种情况下，程序员可以选择他喜欢的方式。

例 3.4

给出等级分数

假设我们要编写一个程序，输入一个数值分数，输出等级分数，即是 A 级，B 级和 C 级

```
      grade > 95      A
95 ≥ grade > 86      B
86 ≥ grade > 76      C
76 ≥ grade > 66      D
66 ≥ grade >   0      F
```

用两种方式写出这个程序，第一种方式用多个 elseif 语句，第二种方式用 if 的嵌套。

答案:

(a)用多个 elseif 语句

```
if grade > 95.0
    disp('The grade is A.');
elseif grade > 86.0
    disp('The grade is B.');
elseif grade >76.0
    disp('The grade is C.');
elseif grade > 66.0
    disp('The grade is D.');
else
    disp('The grade is F.');
end
```

(b)用 if 嵌套结构

```
if grade > 95.0
    disp('The grade is A.');
else
    if grade > 86.0
        disp('The grade is B.');
    else
        if grade > 76.0
            disp('The grade is C.');
        else
            if grade > 66.0
                disp('The grade is D.');
            else
                disp('The grade is F.');
            end
        end
    end
end
```

从上面的例子中，我们可以看到如果有多个选项的话，在一个 if 结构中用到多个 else if 语句将会比 if 的嵌套结构简单的多。

好的编程习惯

对于有许多选项的选择结构来说，最好在一个 if 结构中使用多个 elseif 语句，尽量不用 if 的嵌套结构。

3.4.4 switch 结构

switch 结构是另一种形式的选择结构。程序员可以根据一个单精度整形数，字符或逻辑表达式的值来选择执行特定的代码语句块。

```
switch (switch_expr)
case case_expr_1,
     Statement 1 ⎫
     Statement 2 ⎬ Block 1
     ...         ⎭
case case_expr_2
     Statement 1 ⎫
     Statement 2 ⎬ Block 2
     ...         ⎭
...
otherwise,
     Statement 1 ⎫
     Statement 2 ⎬ Block n
     ...         ⎭
end
```

如果 switch_expr 的值与 case_expr_1 相符，那么第一个代码块将会被执行，然后程序将会跳到 switch 结构后的第一个语句。如果 switch_expr 的值与 case_expr_2 相符，那么第二个代码块将会被执行，然后程序将会跳到 switch 结构后的第一个语句。在这个结构中，用相同的方法来对待其他的情况。otherwise 语句块是可选的。如果它存在的话，当 switch_expr 的值与其他所有的选项都不相符时，这个语句块将会被执行。如果它不存在，且 witch_expr 的值与其他所有的选项都不相符，那么这个结构中的任何一个语句块都不会被执行。这种情况下的结果可以看作没有选择结构，直接执行 **MATLAB** 语言。

如果说 switch_expr 有很多值可以导致相同代码的执行，那么这些值可以括在同一括号内，如下所示。如果这个 switch 表达式和表中任何一个表达式相匹配，那么这个语句块将会被执行。

```
switch (switch_expr)
case {case_expr_1, case_expr_2, case_expr_3},
     Statement 1 ⎫
     Statement 2 ⎬ Block 1
     ...         ⎭
otherwise,
     Statement 1 ⎫
     Statement 2 ⎬ Block n
     ...         ⎭
end
```

switch_expr 和每一个 case_expr 既可以是数值，也可以是字符值。

注意在大多情况下只有一个语句块会被执行。当一个语句块被执行后，编译器就会跳到 end 语句后的第一个语句开始执行。如果 switch 表达和多个 case 表达式相对应，那么只有他们中的第一个将会被执行。

让我们看一个简单的关于 switch 结构的例子。下面的语句用来判断 1 到 10 之间的数是奇数还是偶数。它用来说明一系列的 case 选项值的应用和 otherwise 语块的应用。

```
switch (value)
```

```
case {1, 3, 5, 7, 9},
    disp('The value is odd.');
case {2, 4, 6, 8, 10},
    disp('The value is even.');
otherwise,
    disp('The value is out of range.');
end
```

3.4.5 try/catch 结构的应用

try/catch 结构是选择结构的一种特殊形式，用于捕捉错误。一般地，当一个 **MATLAB** 程序在运行时遇到了一个错误，这个程序就会中止执行。try/catch 结构修改了这个默认行为。如果一个错误发生在这个结构的 try 语句块中，那么程序将会执行 catch 语句块，程序将不会中断。它将帮助程序员控制程序中的错误，而不用使程序中断。

Try/catch 结构的基本形式如下:

```
try
    Statement 1 ┐
    Statement 2 ├ Try Block
    ...         ┘
catch
    Statement 1 ┐
    Statement 2 ├ Catch Block
    ...         ┘
end
```

当程序运行到 try/catch 语句块，在 try 语句块中的一些语句将会被执行。如果没有错误出现，catch 语句块将会被跳过。另一方面，如果错误发生在一个 try 语句块，那么程序将中止执行 try 语句块，并立即执行 catch 语句块。

下面有一个包含 try/catch 结构程序。它能创建一个数组，并询问用户显示数组中的哪能一个元素。用户提供一个下标，那么这个程序将会显示对应的数组元素 try 语句块一般会在这个程序中执行，只有当 try 语句块执行出错，catch 语句块将会发生错误。

```
% Initialize array
a = [ 1 -3 2 5];
try
    % Try to display an element
    index = input('Enter subscript of element to display: ');
    disp(['a(' int2str(index) ') = ' num2str(a(index))] );
catch
    % If we get here an error occurred
    disp( ['Illegal subscript: ' int2str(index)] );
end
```

这个程序的执行结果如下：

```
>> try_catch
Enter subscript of element to display: 3
a(3) = 2
>> try_catch
Enter subscript of element to display: 8
Illegal subscript: 8
```

测试 3.2

本测试提供了一个快速的检查方式，看你是否掌握了 3.4 的基本内容。如果你对本测试有疑问，你可以重读 3.4，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

根据下面的描述编写对应的 **MATLAB** 语句。

1. 如果 x 大于等于 0，把 x 的平方根赋值于变量 sqrt_x，并打印出结果。否则，打印出一条关于平方根函数参数的错误信息。并把 sqrt_x 归零。

2. 变量 fun 由 n/m 计算得到，如果 m 的绝对值小于 $1.0e^{-300}$，打印出除数为 0，否则计算并打印出 fun 值。

3. 租用一个交通工具前 100 公里 0.50 美元每公里，在下面的 200 公里中 2.30 美元每分钟，越过 300 公里的部分一律按 0.20 美元每公里。已知公里数，编写对应的 **MATLAB** 语句计算出总费用和平均每公里的费用。

检测下面的 **MATLAB** 语句，是对是错?正确的，输出结果如何，错误的，错在哪里?

4.
```
if volts > 125
   disp('WARNING: High voltage on line.');
   if volts < 105
      disp('WARNING: Low voltage on line.');
   else
      disp('Line voltage is within tolerances.');
   end
```

5.
```
color = 'yellow';
switch( color);
case 'red',
    disp('Stop now!');
case 'yellow',
    disp('Prepare to stop.');
case 'green',
    disp('Proceed through intersection.');
otherwise,
    disp('Illegal color encountered.');
end
```

6.
```
if temperature > 37
   disp('Human body temperature exceeded.');
elseif temperature > 100
   disp('Boiling point of water exceeded.');
end
```

3.5 附加的画图特性

在本节中，我们将讨论简单的二维图象(在第二章我们已有所介绍)的附加特性。这些特性将允许我们控制 x，y 轴上的值的范围，在一个坐标系内打印多个图象，或创建多个图，或在一个图象窗口内创建多个子图像，或提供更加强大的轨迹文本字符控制。还有，我们将向大家如何创建极坐标。

3.5.1 控制 x，y 轴绘图的上下限

在默认的情况下，图象的 X，Y 轴的范围宽到能显示输入值的每一个点。但是有时只显

示这些数据的一部分非常有用，这时你可以应用 axis 命令/函数。

axis 命令/函数的一些形式展示在表 3.5 中。其中两个最重要的形式在表中用黑体字标出——它允许程序员设定和修改坐标的上下限。所有形式的完全列表将会在 **MATLAB** 的在线文件中找到。

为了说明 axis 的应用，我们将画出函数 f(x)=sinx 从-2π 到 2π 之间的图象，然后限定坐标的区域为 0≤x≤π，0≤y≤1。

表 3.5axis 函数/命令的形式

命令	功能
v=axis	此函数将会返回一个 4 元素行向量[xmin xmax ymin ymax]，其中 xmin xmax ymin ymax 代表 x，y 轴的上下限
axis([xmin xmax ymin ymax])	xmin xmax 设定横轴的下限及上限， ymin ymax 设定纵轴的下限及上限
axis equal	将横轴纵轴的尺度比例设成相同值
axis square	横轴及纵轴比例是 1:1
axis normal	以预设值画纵轴及横轴
axis off	将纵轴及横轴取消
axis on	这个命令打开所有的轴标签，核对符号，背景(默认情形)

一些 **MATLAB** 命令似乎不能确定它是个函数还是一个命令。例如，有时 axis 它好像是命令，有时它好像是函数。有时我们把它当作命令:axis on，在其他时候，我们把他当作函数:axis([0 20 0 35])。遇到这样的情况怎么办?

一个简单的答案是 **MATLAB** 命令是通过函数来实现的。MALTAB 编译器无论什么时候遇到这个命令，它都能转化为相应的函数。它把命令直接当作函数来用，而不是应用命令语法。下面的两个语句是等价的：

```
axis on;
axis ('on');
```

无论什么时候 **MATLAB** 遇到一个命令时，它都会转化一个函数，当命令的参数当作字符串看作相对应函数的参数。所以编译器翻译如下命令:

```
garbage 1 2 3
```

为

```
garbage ('1', '2', '3')
```

注意只有带有字符参数的函数才能当作命令。带有数字参数的函数只能被当作函数。这就是为什么 axis 有时当作命令，有时被当作函数。

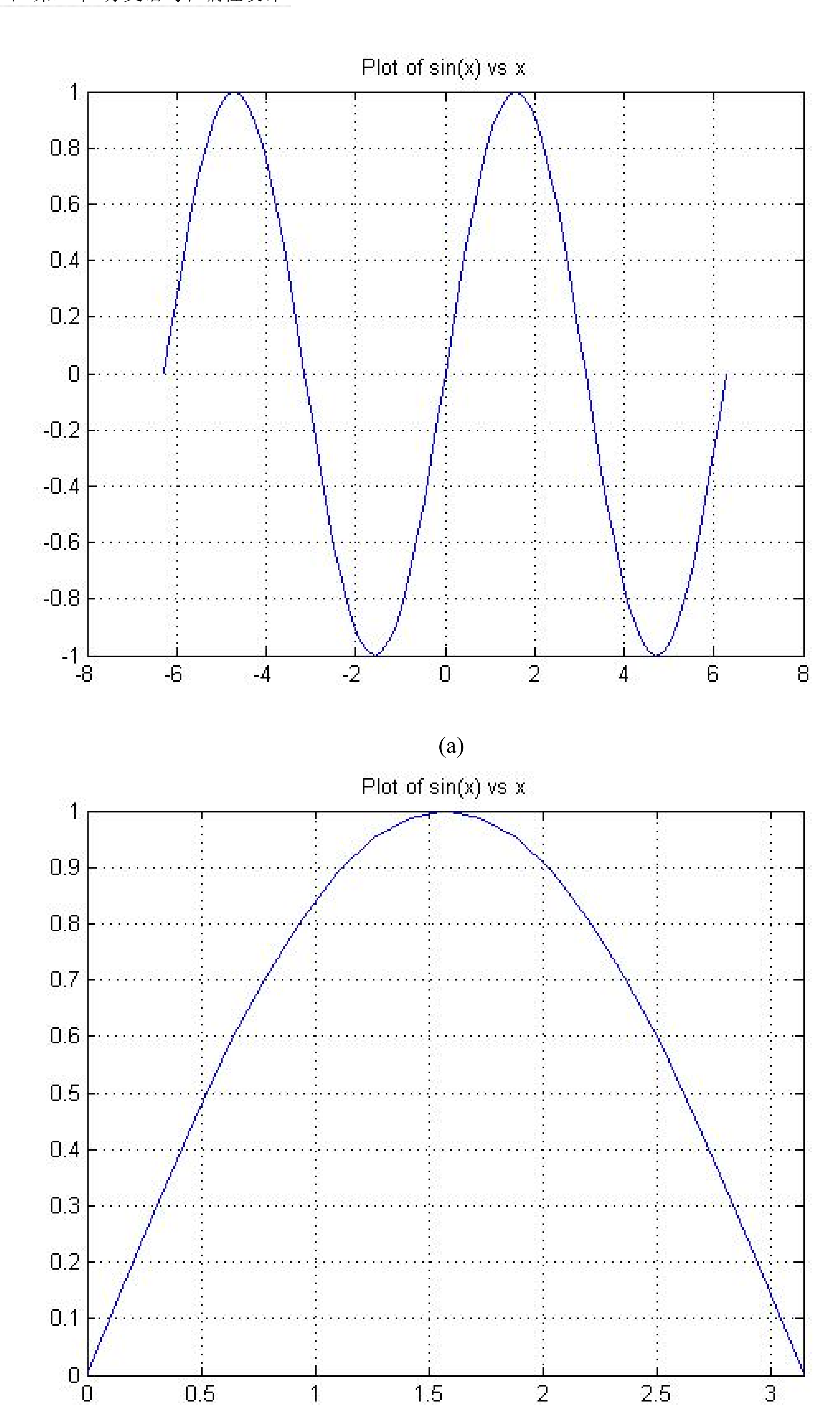
```
x=-2*pi:pi/20:2*pi;
y=sin(x);
plot(x,y);
title('Plot of sin(x) vs x');
```

当前图象坐标轴的上下限的大小由函数 axis 得到。

```
>> limits=axis
limits =
     -8     8    -1     1
```

修改坐标轴的上下限可以调用函数 axis([0 pi 0 1])。

当这个函数执行后，产生的图象如图 3.3（b）所示。



(b)

图 3.3 (a)以 x 为自变量的 sinx 的图象 （b）画图区域为 [0 π 0 1]

3.5.2 在同一坐标系内画出多个图象

在一般情况下，创建一个新的图象就要用到一个 plot 命令，前面的数据就会自动消失。这种行为可以通过使用 hold 命令得到修改。当 hold on 命令执行后，所有的新的图象都会叠加在原来存在的图象。hold off 命令可恢复默认情况，用新的图象来替代原来的图象。

例如，在同一坐标轴内的画出 sinx 和 cosx 的图象。产生的图象如图 3.4 所示。

```
x = -pi:pi/20:pi;
y1 = sin(x);
y2 = cos(x);
plot(x,y1,'b-');
hold on;
plot(x,y2,'k--');
hold off;
legend ('sin x','cos x');
```

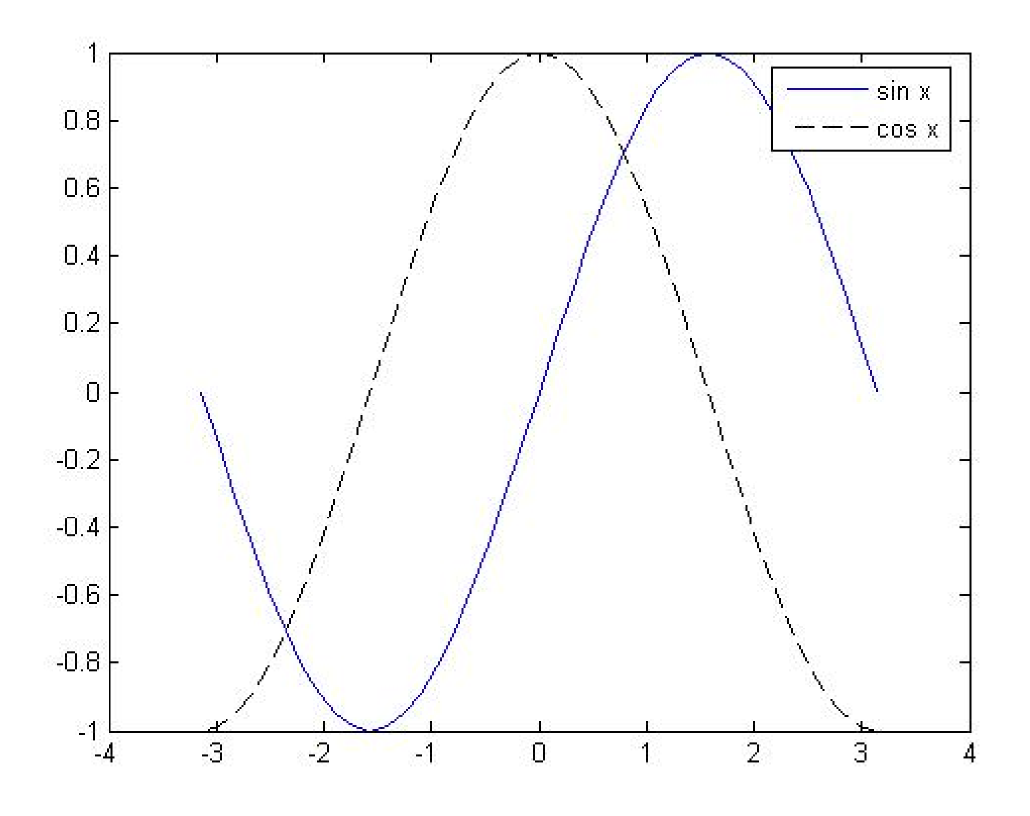


图 3.4 用 hold 命令在一个坐标轴内画出两个函数的图象

3.5.3 创建多个图象

MATLAB 可以创建多个图象窗口，每个窗口都有不同的数据。我们用**图象数**来区分这些图象窗口，**图象数**是一个小的正整数。第一个图象窗口称为图 1，第二个图象窗口为图 2，依次类推。这些窗口中的一个称为当前图象窗口，所有的新的画图命令将会展示在那个窗口中。

我们用 figure 函数来选择当前窗口。这个函数的形式为“figure(n)”，其中 n 代表图象数。当这个函数被执行后，图 n 将会变为当前图象，执行所有的画图命令。如果这个图象窗口不

存在，那么 **MATLAB** 将会自动创建。当前图象也可以用鼠标单击选择。

gcf 函数用于当于返回当前图象数。当你需要知道当前图象数时，你就把这个函数写入 M 文件中。

下面的命令用于说明图函数的应用。它将创建两个图象，第一个用来展示 e^x 的图象，第二个用来展示 e^{-x} 的图象。

```
figure(1);
x = x:0.05:2;
y1 = exp(x);
plot(x,y1);
figure(2);
y2 = exp(-x);
plot(x,y2);
```

3.5.4 子图象

在一个图象窗口中有一系列的坐标系，创建出多个子图象。创建子图象要用到 subplot 命令其形式如下

```
subplot(m,n,p)
```

这个命令在当前图象窗口创建了 m×n 个子图象，按 m 行，n 列排列，并选择子图象 p 来接受当前所有画图命令。

这些子图象以从左向右从上到下编号。例如，命令 subplot(2,3,4)将会创建 6 个子图象，而且 subplot 4 是当前子图象。

如果 subplot 命令创建的新坐标系与原来的坐标系相冲突，那么原来的坐标系将会被自动删除。

下面的命令将会在同一窗口中创建两个子图象，每一个子图象独立地展示不同的图象。产生的图象为图 3.5。

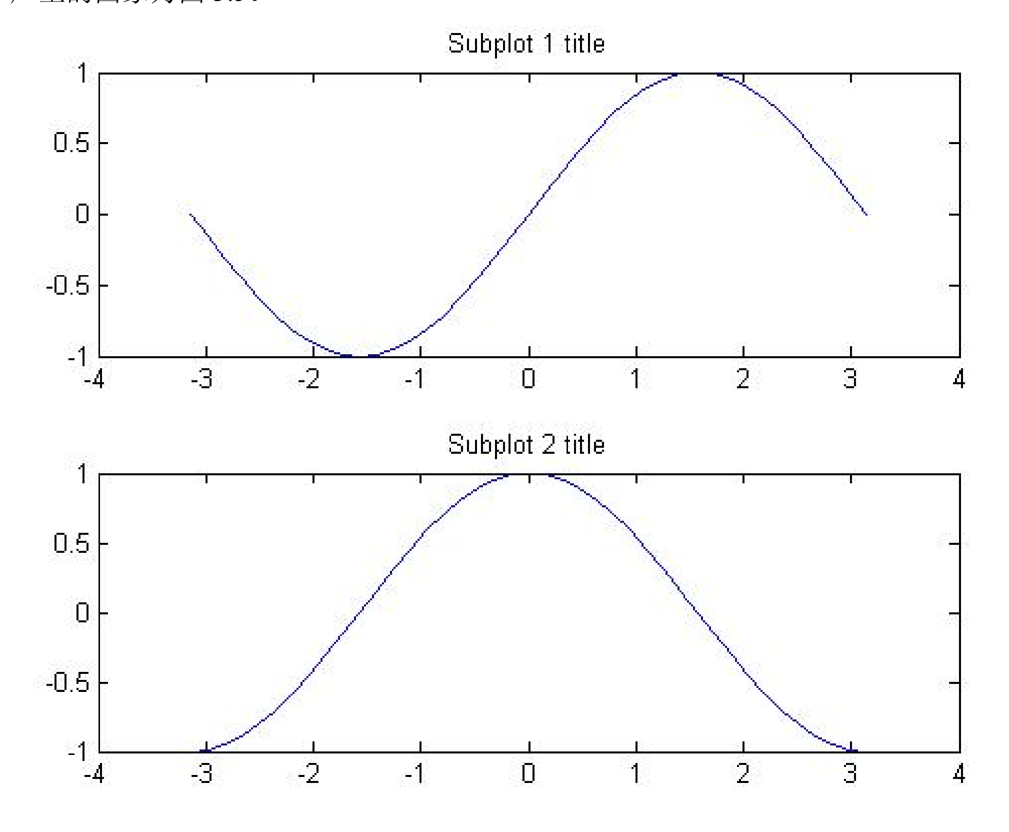


图 3.5 包含多个子图象的图象

```
figure(1);
subplot(2,1,1);
x = -pi:pi/20:pi;
y = sin(x);
plot(x,y);
title('Subplot 1 title');
subplot(2,1,2);
x = -pi:pi/20:pi;
y = cos(x);
plot(x,y);
title('Subplot 2 title');
```

3.5.5 对画线的增强控制

在第二章中，我们学习了如何设置画线的颜色，样式，符号形式。我们还可以设置其中的 4 种附加的属性。

属性	说明
LineWidth	用来指定线的宽度
MarkerEdgeColor	用来指定标识表面的颜色
MarkerFaceColor	填充标识的颜色
MarkerSize	指定标识的大小

在 plot 命令中，在自变量和函数之后被指定，形式如下：

```
plot(x,y,'PropertyName',value,...)
```

例如，下面的命令将画出一个图象，轨迹的宽度为 3，颜色为黑色，圆圈标识的宽度为 6，每个标识为红色边缘和绿色内核，如图 3.6。

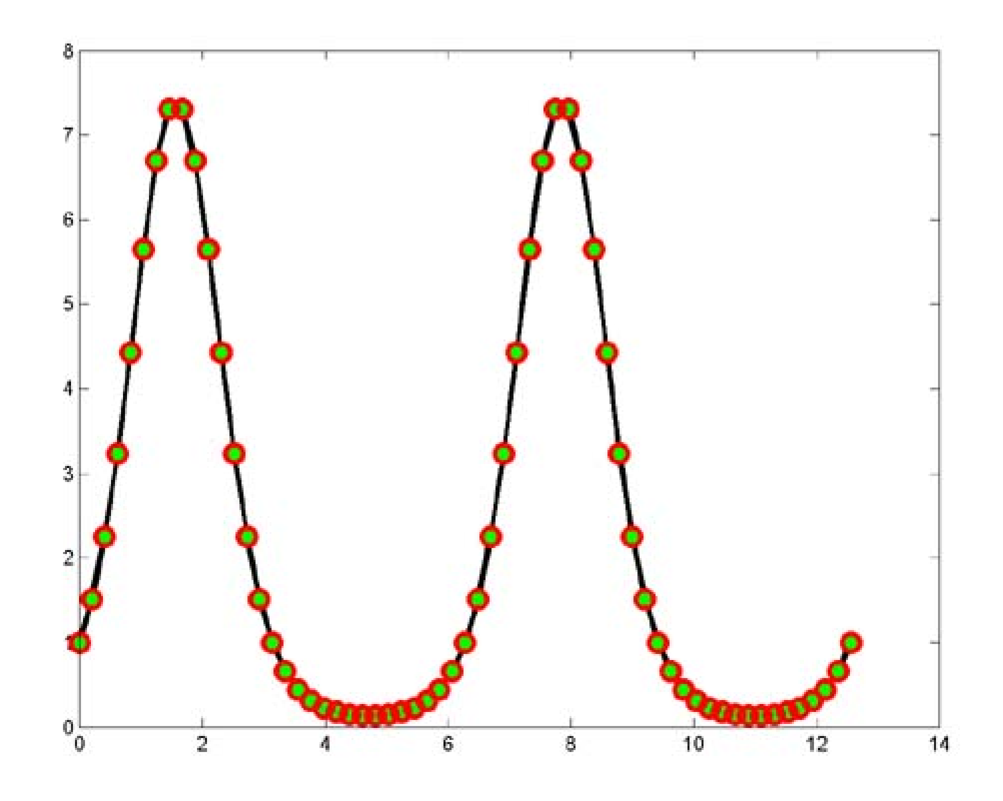


图 3.6 这个图象用于说明 LineWidth 和 Marker 的属性

3.5.6 文本字符串的高级控制

我们在画图中可能要用到文本字符串(比如标题，坐标轴标签)，这些字符串我们可以用黑体，斜体来格式化，也包括特殊的希腊或数学符号。

文本的字体通可以通过 stream modifiers 修改。一个 stream modifier 是一个特殊的字符序列，

用来告诉编译器改变它的行为。最普通的 stream modifiers 是:

\bf	黑体
\it	斜体
\rm	恢复正常字体
\fontname	字体的名字
\fontsize	字体的大小
_{xxx}	xxx 做为某字符的上标
^{xxx}	xxx 做为某字符的下标

一旦一个 stream modifier 插入一个文本字符串中，它持续发挥作用，直到这个字符串的结束或消失。如果一个 modifier 后在跟着一个{}，只有{}中的文本起作用。

特殊的希腊字母或数学符号也可用在文本字符串中。通过嵌入特殊的转义序列来创建这些字符。这些转义序列是支持 T_EX 语言的特殊序列的一个子集。在表 3.6 中向大家展示一些转义序列代码的例子。所有转义序列可以在 **MATLAB** 在线帮助文本中找到。

如果要打印转义符\，{，}，_，或^就必须在前面加上一个反斜杠。

下面的例子用于说明 stream modifier 和特殊字符的应用。

字符串	结果
\tau_{ind} versus \omega_{\itm}	τ_{ind} versus ω_{m}

\theta varies from 0\circ to 90\circ	θ varies from 0° to 90°
\bf{B}_{\itS}	$\mathbf{B}_{S}$

表 3.6 精选的希腊符号和数学符号

字符序列	符号	字符序列	符号	字符序列	符号
\alpha	α			\int	$\int$
\beta	β			\cong	$\cong$
\gamma	γ	\Gamma	Γ	\sim	$\sim$
\delta	δ	\Delta	Δ	\infty	∞
\epsilon	ε			\pm	$\pm$
\eta	η			\leq	$\leq$
\theta	θ			\geq	$\geq$
\lamda	λ	\Lamda	Λ	\neq	$\neq$
\mu	μ			\propto	$\propto$
\nu	ν			\div	$\div$
\pi	π	\Pi	Π	\circ	$\circ$
\phi	φ			\leftritharrow	$\leftrightarrow$
\rho	ρ			\leftarrow	$\leftarrow$
\sigma	σ	\Sigma	Σ	\rightarrow	$\rightarrow$
\tau	τ			\uparrow	$\uparrow$
\omega	ω	\Omega	Ω	\downarrow	$\downarrow$

3.5.7 极坐标图象

MATLAB 中包括一个重要的函数叫做 polar，它用于在极坐标系中画图。这个函数的基本形式如下:

```
polar(theta,r)
```

其是 theta 代表一个弧度角数组，r 代表一个距离数组。它用来画以角度为自变量的函数的极坐标图是非常有用的。

例 3.5

心形麦克风

为舞台表演设计的麦克风大多都是定向麦克风，它能够增大来自演唱者的信号，抑制后面观众的噪声信号。一个心形麦克风的增益 gain 是关于角度 θ 的函数，关系式如下

$$Gain=2g(1+\cos\theta) \qquad (3.3)$$

其中 g 是和特定的心形麦克风有关的常量。θ 是声源和麦克风之间的夹角。假设一个麦克风的 g 是 *0.5*，画出函数 *Gain* 的极坐标图。

答案：我们必须计算出与角度对应的函数值，然后画出相应的极坐标图。产生的结果如图 3.7 所示。注意这种麦克风叫做心形麦克风，所以得出来曲线的形状像颗心。



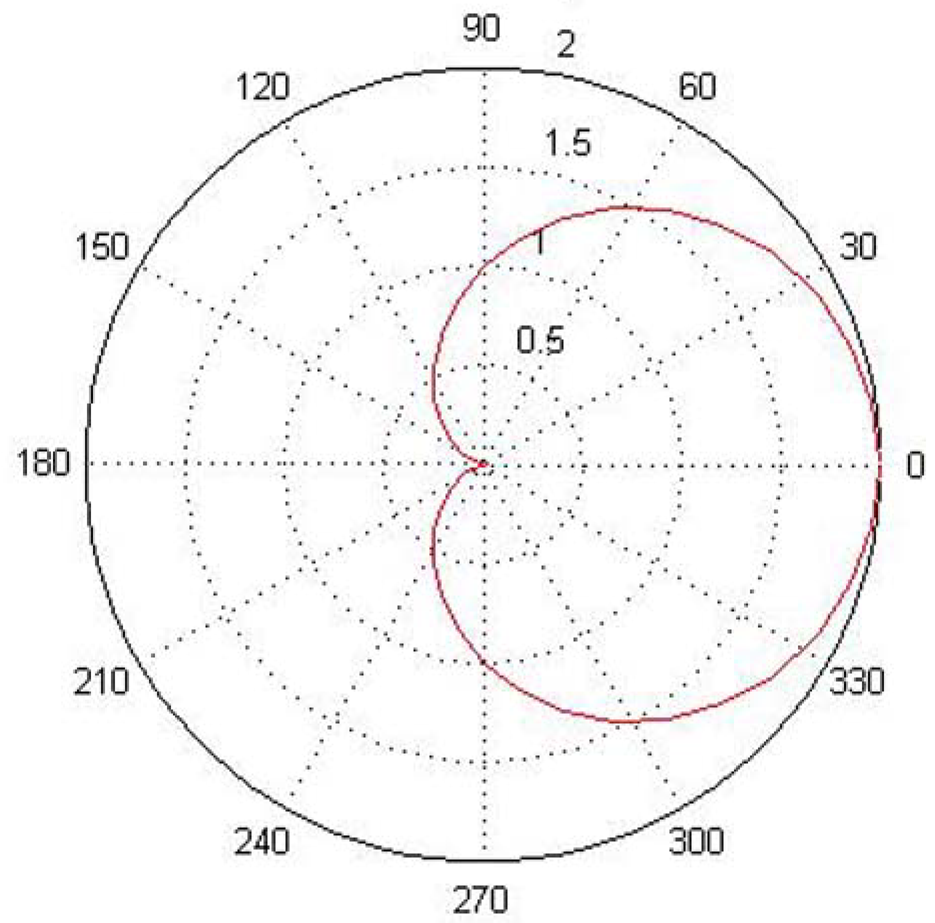


图 3.7 心形麦克风的增益图象

代码如下：

```
% Script file: microphone.m
%
% Purpose:
% This program plots the gain pattern of a cardioid
% microphone.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/10/97 S. J. Chapman Original code
%
% Define variables:
% g -- Microphone gain constant
% gain -- Gain as a function of angle
% theta -- Angle from microphone axis (radians)
% Calculate gain versus angle
g = 0.5;
theta = 0:pi/20:2*pi;
gain = 2*g*(1+cos(theta));
% Plot gain
polar (theta,gain,'r-');
title ('Gain versus angle \it\theta');
```

例 3.6

电器工程低通滤波电路

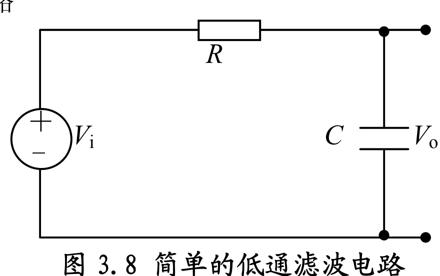


图 3.8 简单的低通滤波电路

上图是向大家展示的一个简单的低通滤波电路。这个电路是由一个电阻和一个电容组成。输出电压 V_0 与输入电压 V_i 的电压比为

$$\frac{V_o}{V_i}=\frac{1}{1+j2\pi fRC} \tag{3.4}$$

其中 V_i 是在频率 f 下的正弦输入电压。R 代表电阻，单位为欧姆。C 代表电容，单位为法拉。j 为$\sqrt{-1}$

假设 R=16 kΩ，电容 C=1 μF，画出这个滤波器，振幅与频率的关系图。由于频率和振幅的关系图两者的跨度都非常的大，按照惯例，两者均使用对数标度，另外相位的取值范围非常的小，所以对相位我们应用线性标度。

所以，我们将用 loglog 命令来画频率响应，用 semilogx 来画相位响应图。我们将在一个画图窗口内画出两个子图象。

代码如下：

```
% Script file: plot_filter.m
%
% Purpose:
% This program plots the amplitude and phase responses
% of a low-padd RC filter.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/29/98 S. J. Chapman Original code
%
% Define variables:
% amp     -- Amplitude response
% C       -- Capacitiance (farads)
% f  -- Frequency of input signal (Hz)
% phase   -- Phase response
% R       -- Resistance (ohms)
% res     -- Vo/Vi
% Initialize R & C
R = 16000; % 16 k ohms
C = 1.0E-6; % 1 uF
% Create array of input frequencies
f = 1:2:1000;
% Calculate response
res = 1 ./ ( 1 + j*2*pi*f*R*C );
% Calculate amplitude response
amp = abs(res);
% Calculate phase response
phase = angle(res);
```

```
% Create plots
subplot(2,1,1);
loglog( f, amp );
title('Amplitude Response');
xlabel('Frequency (Hz)');
ylabel('Output/Input Ratio');
grid on;
subplot(2,1,2);
semilogx( f, phase );
title('Phase Response');
xlabel('Frequency (Hz)');
ylabel('Output-Input Phase (rad)');
grid on;
```

得到的结果如图 3.9 所示。注意这个电路叫做低通滤波电路，是因为在低频下，电压很少衰减，在高频下，电压衰减的很厉害。

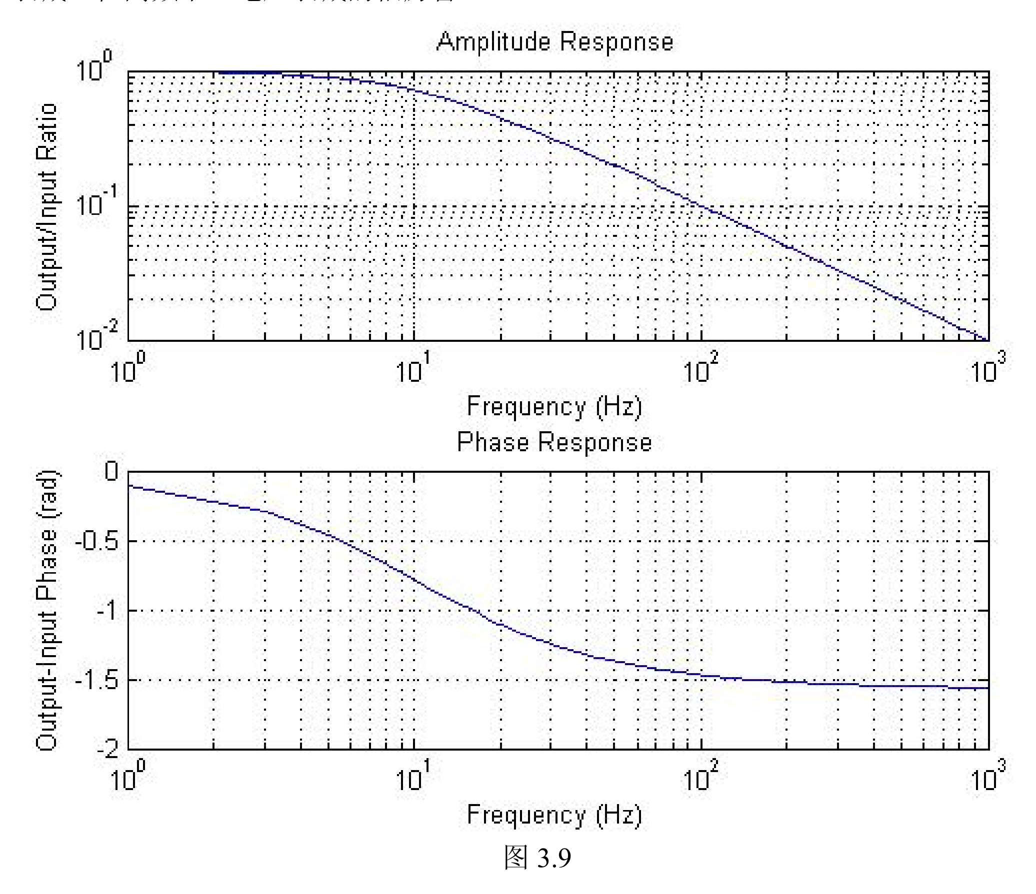


图 3.9

例 3.7

热力学：理想气体定律

理想气体是指发生在分子之间的碰撞均为弹性碰撞。你可以把理想气体中的每一个分子想象成一个刚性小弹，每次碰撞，总动能不会改变。这样的气体可以用三个变量来描述：绝对气压（P），体积（V）和绝对温度（T）。三者之间的关系式就是我们所熟知的理想气体定律。

$$PV=nRT \tag{3.5}$$

P 代表气压，单位为千帕，V 代表气体的体积，单位为升，n 代表分子的摩尔数，T 代表绝对温度，单位为开。

假设一理想气体样品在 273K 温度下，有一摩尔分子，回答相关问题。

（a）当气压从 1 到 1000 千帕变化，气体的体积将会如何变化？设置合适的坐标，画出这个气体的压力——体积图象。

（b）假设这个气体的温度上升到 373K，气体体积将会随气压如何变化。在与（a）相同的坐标系内，画出气体的压力——体积图象。轨迹用虚绿线，宽度为 2pixel。在图象上包含有一个大标题，x，y 轴的标签，还有各轨迹的图例。

答案：因为我们画的值都有一千个因子，所以一个普通线性尺度坐标不能画出有效的图象。所以，我们在画图时，用 log-log 标度。注意我们必须在相同的坐标系下，画出两个曲线，所以我们必须在画完第一个图象后加入 hold on 命令，当所有画图结束后，用上 hold off 命令。我们也必须指定轨迹的颜色，样式和宽度，并指定标签为黑体。

下面的程序创建了气压的函数 *V*(气体的体积)的图象。注意那些控制图象样式的语句，我们已用黑体标出。

```
% Script file: ideal_gas.m
%
% Purpose:
%     This program plots the pressure versus volumn of an
%     ideal gas.
%
% Record of revisions:
%     Date          Programmer        Description of change
%     ====          ==========        =====================
%     07/17/00      S.J.Chapman       Original code
%
% Define variables:
% n             --Number of atoms (mol)
% P             --Pressure (kPa)
% R             --Ideal gas constant (L kPa/mol K)
% T             --Temperature (K)
% V             --volume (L)
% Initialize nRT
n = 1;            % Moles of atoms
R = 9.314;        % Ideal gas constant
T = 273;          % Temperature (K)
% Create array of input pressures. Note that this
% array must be guite dense to catch the major
% changes in volume at low pressures.
P = 1:0.1:1000;
% Calculate volumes
V = (n * R * T) ./ P;
% Create first plot.
figure(1);
loglog(P, V, 'r-', 'LineWidth', 2);
title('\bfVolume vs Pressure in an Ideal Gas');
xlabel('\bfPressure (kPa)');
ylabel('bfVolume (L)');
grid on;
hold on;
% Now increase temperature
T = 373;          % Temperature (K)
% Calculate volumes
V = (n * R * T) ./ P;
% Add second line to plot
figure(1);
loglog(P, V, 'b--', 'LineWidth', 2);
hold off;
% Add legend
legend('T = 273 K', 'T = 373 K');
```

产生的 V-P 图象如图 3.10 所示。

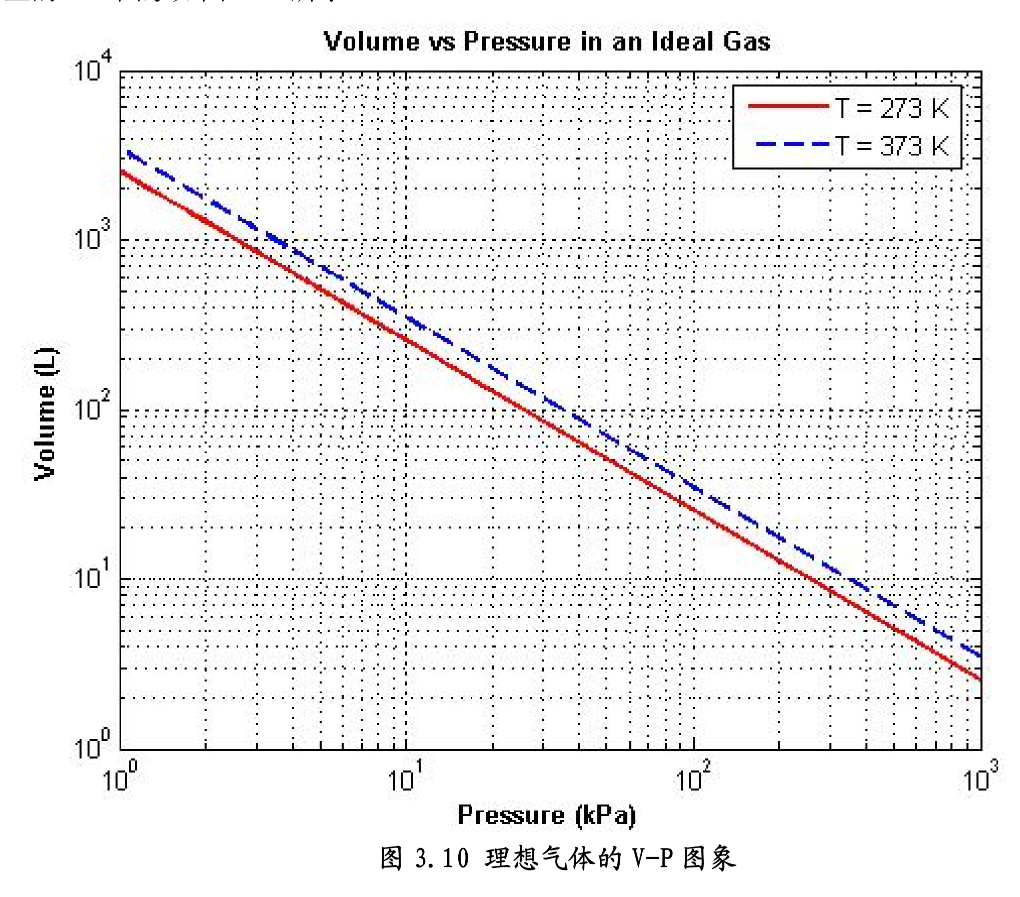


图 3.10 理想气体的 V-P 图象

3.5.8 注释并保存图象

一旦 **MATLAB** 成功创建一个图象，那么用户就可以运画图工具条上的 GUI 工具来编辑和注释这些图象。图 3.11 向大家展示了这些可用的工具，它允许我们添加直线，带箭头的线，还有文本。当工具条中的编辑按钮()被选中，注释和翻译工具将会变得可用。

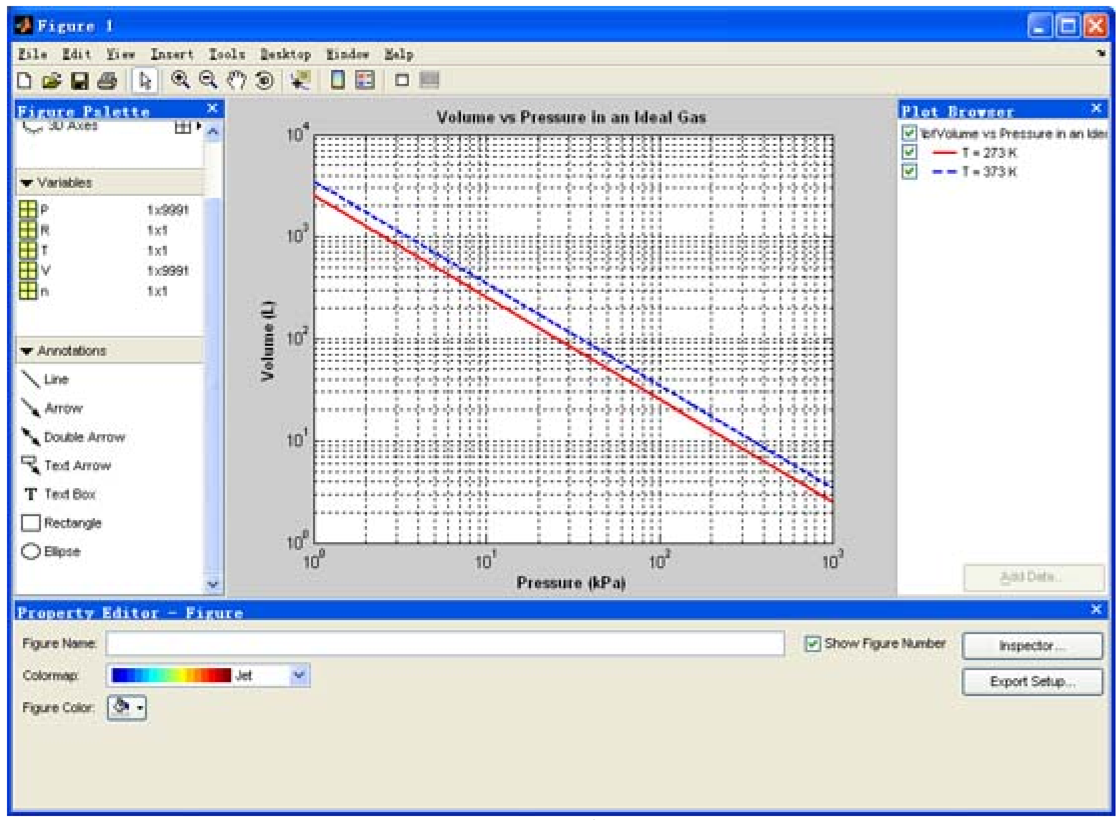


图 3.11 图工具条中的编辑工具

还有，当编辑按钮被按下，单击图象中的任何一条线或一个文本，它们将会处于可编辑状态，如果双击它们将会弹出一个属性窗口，允许我们修改这个对象的每一项属性。图 3.12 是图 3.10 经用户编辑修改后得到的，用户把绿线改成了 3pixel 宽的虚线，并加上了箭头和注释。

当这个图象的编辑和注释完成后，你可以一种可修改的格式存储整个图象，方法选择图象窗口中的”file/save as”菜单项。产生的图象文件(*.fig)包含了用于重建这个图象的所有信息，也就是说在未来的任何时侯你都可以轻松的重建这个图象。

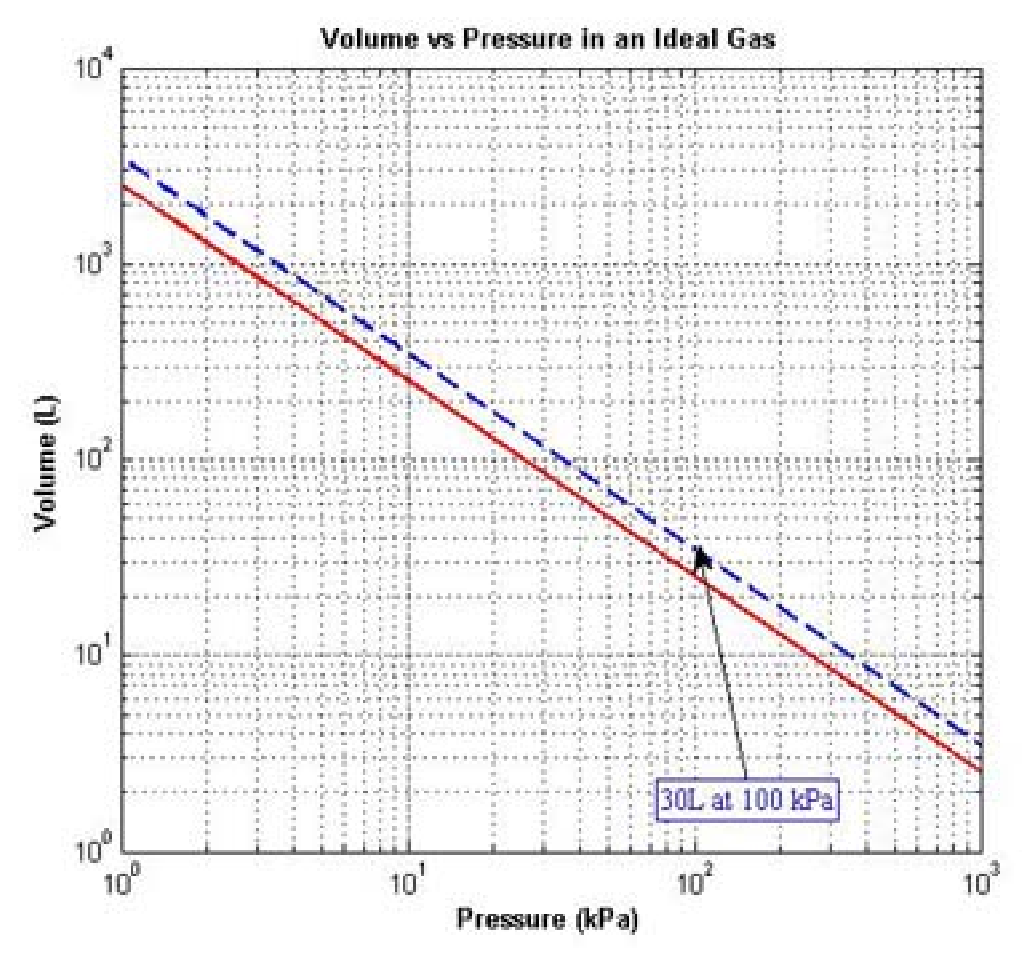


图 3.12 应用图工具条中的纺辑工具改变了蓝线的样式并添加了注释

测试 3.3

本测试提供了一个快速的检查方式，看你是否掌握了 3.5 的基本内容。如果你对本测试有疑问，你可以重读 3.5，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. 编写 **MATLAB** 语句，画出 sinx 和 cos2x 在 0 到 2π 之间的图象，其中步增量为 π/10。这个些点由 2pixel 宽的红线相边，每一个点用 6pixel 宽的绿色圆圈标记。

2. 应用图象编辑工作改变标记符号为黑色方块。添加注释并用带箭头的线指向 x=π 的点。用 **MATLAB** 文本字符编写以下表达式

3. $f(x) = \sin\theta\cos 2\phi$

4. Plot of Σx^2 versus x

下面的文本字符将产生怎样的表达式

5. '\tau\it_{m}'

6. '\bf\itx_{1}^{ 2} + x_{2}^{ 2} \rm(units:\bfm^{2}\rm)'

7. 如何在文本字符串产生反斜杠(\)?

3.6 程序调试的进一步说明

在含有选择结构和循环结构的程序出错的概率要比只含简单的顺序结构的程序出错的概率大得多。在完成了程序设计的步骤之后，无论多大的一个程序，在第一次运行时都很难

通过。假如我们创建了一个程序并调试它，只发现这个程序的输出是错误的。我们怎样找到这些错误并修改它呢?

一旦程序包含了循环和选择结构，找到错语的最好的方法是应用 **MATLAB** 支持的符号调试器(symbolic debugger)。这个调试器将会整合到 **MATLAB** 编辑器中。

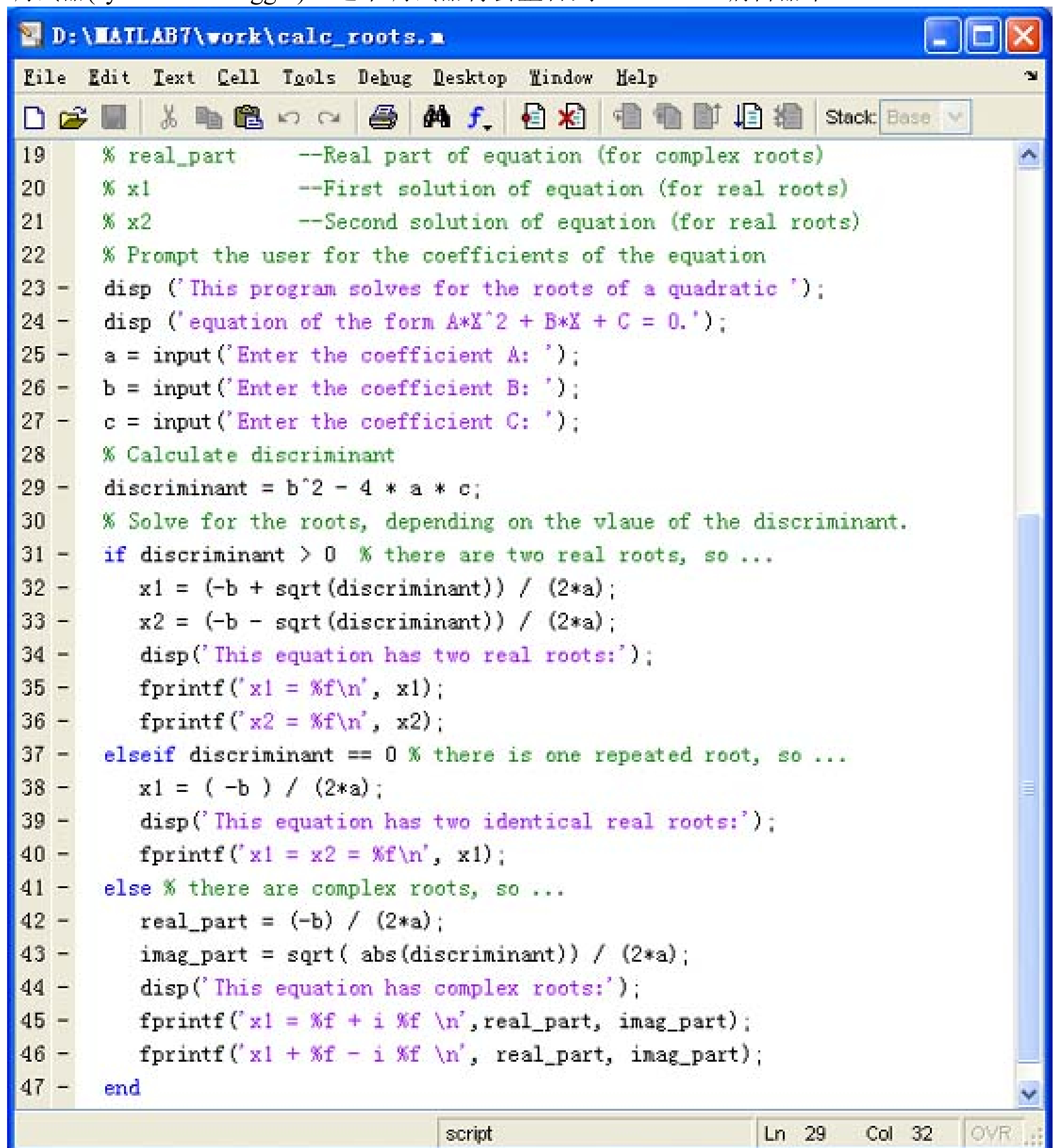


图 3.13 MATLAB 的编辑/调试窗口

应用这个调试器，首先应该选择“file/open”打开你在 **MATLAB** 命令窗口中要调试的程序。当一个文件被打开，编辑器就加载了这个文件，代码根据语法的不同出现不同的颜色。在这个文件中评论显示为绿色，变量和数字显示为黑色，字符串显示为红色，语言的关键字显示为蓝色。图 3.13 向大家展示的是含有文件 calc_roots.m 的编辑/调试窗口。

当一个程序执行时，我们想知道什么事情发生了。为了达到此目的，我们可以用鼠标右击你所关心的行并选择”set/clear breakpoint”选项。当一个断点被设置后，一个红色的点将会出现在行的左边，如图 3.14 所示。

```
D:\MATLAB7\work\calc_roots.m
File Edit Text Cell Tools Debug Desktop Window Help

24 -  disp ('equation of the form A*X^2 + B*X + C = 0.');
25 -  a = input('Enter the coefficient A: ');
26 -  b = input('Enter the coefficient B: ');
27 -  c = input('Enter the coefficient C: ');
28    % Calculate discriminant
29 -  discriminant = b^2 - 4 * a * c;
30    % Solve for the roots, depending on the vlaue of the discriminant.
31 -  if discriminant > 0  % there are two real roots, so ...
32 ●     x1 = (-b + sqrt(discriminant)) / (2*a);
33 ●     x2 = (-b - sqrt(discriminant)) / (2*a);
34 ●     disp('This equation has two real roots:');
35 -     fprintf('x1 = %f\n', x1);
36 -     fprintf('x2 = %f\n', x2);
37 -  elseif discriminant == 0 % there is one repeated root, so ...
38 -     x1 = ( -b ) / (2*a);
39 -     disp('This equation has two identical real roots:');
40 -     fprintf('x1 = x2 = %f\n', x1);
41 -  else % there are complex roots, so ...
42 -     real_part = (-b) / (2*a);
43 -     imag_part = sqrt( abs(discriminant)) / (2*a);
44 -     disp('This equation has complex roots:');
45 -     fprintf('x1 = %f + i %f \n',real_part, imag_part);
46 -     fprintf('x1 + %f - i %f \n', real_part, imag_part);
47 -  end

script    Ln 33   Col 36   OVR
```

图 3.14 这个窗口断点已设置

一旦这些断点被设置，在命令窗口键入 calc_roots 将会像往常一样执行这个程序。这个程序将会运行到第一个断点并在那里停止。在调试的过程中将会有一个绿色的箭头将会出现在当前行。如图 3.15 所示。

一旦到达某个断点程序员可以通过在命令窗口中键入变量名的方法检查或修改在工作区内的任一变量。当程序员对程序的这一点感到满意时，可以通过重复按 F10 一行一行调试，也可以按 F5 运行到下一个断点。它总是能检测程序中的每一个断点中的任何一个变量的值。

调试器的另处一个重要特性是可在 Breakpoints 菜单中找到。这个菜单包括两个项目："stop if Error"和"stop if warning"。如果程序中发生了一个错误，这个错误导致了电脑死机或产生了错误信息，程序员可以打开这些选项，并执行这个程序。这个程序将会运行到错误或警告的断点并停在那儿，它允许程序员检查变量的每一个值，并帮助找出出错的原因。当一个错误被发现，程序员能用编辑器来更正这个 MALTAB 程序，并把更新的版本存到磁盘上，在调试没结束之前，它必须重复以上的动作。这个步骤将会重复下去直到这个程序没有错误出错。

现在花一定的时间来熟悉这个调试器——这是值得的。

```
D:\MATLAB7\work\calc_roots.m
File Edit Text Cell Tools Debug Desktop Window Help
Stack: Base

23 -  disp ('This program solves for the roots of a quadratic ');
24 -  disp ('equation of the form A*X^2 + B*X + C = 0.');
25 -  a = input('Enter the coefficient A: ');
26 -  b = input('Enter the coefficient B: ');
27 -  c = input('Enter the coefficient C: ');
28    % Calculate discriminant
29 -  discriminant = b^2 - 4 * a * c;
30    % Solve for the roots, depending on the vlaue of the discriminant.
31 -  if discriminant > 0  % there are two real roots, so ...
32 -     x1 = (-b + sqrt(discriminant)) / (2*a);
33 -     x2 = (-b - sqrt(discriminant)) / (2*a);
34 -     disp('This equation has two real roots:');
35 -     fprintf('x1 = %f\n', x1);
36 -     fprintf('x2 = %f\n', x2);
37 -  elseif discriminant == 0 % there is one repeated root, so ...
38 -     x1 = ( -b ) / (2*a);
39 -     disp('This equation has two identical real roots:');
40 -     fprintf('x1 = x2 = %f\n', x1);
41 -  else % there are complex roots, so ...
42 -     real_part = (-b) / (2*a);
43 -     imag_part = sqrt( abs(discriminant)) / (2*a);
44 -     disp('This equation has complex roots:');
45 -     fprintf('x1 = %f + i %f \n',real_part, imag_part);
46 -     fprintf('x1 + %f - i %f \n', real_part, imag_part);
47 -  end
48

script    Ln 15    Col 48    OVR
```

图 3.15 在调试的过程中一个绿色的小箭头将会出现在当前行的左侧

3.7 总结

在本章中，我们向大家展示了基本的 **MATLAB** 选择结构，还有控制这个结构的关系运算符和逻辑运算符。这个结构的基本类型是 if 结构。这个结构非常的灵活。如果这个结构需要的话，它可以跟任意多个 elseif 语句，if 结构可以进行嵌套组成更复杂的结构。第二种选择结构是 switch 结构，它提供多项选择。第三种选择结构是 try/catch 结构。它用于跳过错误以保证程序的继续进行。

第三章我们向大家介绍了更多的画图方法。axis 命令允许程序员指定 X，Y 轴的取值范围。hold 命令允许程序员把后面的图象叠加到原来的图象上打印。图命令允许程序员创建和选择多个图象窗口。subplot 命令允许程序在一个图象窗中创建多个子图象。

还有，我们学习如何控制画图的附加功能，例如线的宽度和符号的颜色。这些属性可由指定的“propertyname”和值 Value 决定，“propertyname”和值 Value 将出现在 plot 命令的数据后。运用流编辑器和转义序列将会增强对文本字符串的控制。用流字符串允许程序员指定相应的特性，例如字符的粗斜体，上下标和字体大小，字体类别。我们可以应用转义序列允许在文本中加入特殊的字符，比如说希腊字符和数学符号。

3.7.1 好的编程习惯的总结

在有选择结构和循环结构的编程中，要遵循以下的编程指导思想。如果你长期坚持这些原则，

你的代码将会有很少的错误，有了错误也易于修改，而且在以后修改程序时，也使别人易于理解。

1．在我们检测两数值是否相等时一定要小心，因为 round off 错误可能会使两个本来应该相等的值不相等了。这时你可以在 round off 错误的范围内它是不是近似相等。

2．遵守基本编程设计步骤来编写可靠，易理解的 **MATLAB** 的程序。

3．在 if 结构和 switch 语句中，语句块要缩进两个空格

3.7.2 MATLAB 总结

下面的总结列举了本章出现的所有特殊符号，命令和函数，后面跟的是简短的描述。

v=axis	此函数将会返回一个 4 元素行向量[xmin xmax ymin ymax]，其中 xmin xmax ymin ymax 代表 x，y 轴的上下限
axis([xmin xmax ymin ymax])	以 xmin xmax 设定横轴的下限及上限，以 ymin ymax 设定纵轴的下限及上限
axis equal	将横轴纵轴的尺度比例设成相同值
axis square	横轴及纵轴比例是 1:1
axis normal	以预设值画纵轴及横轴
axis off	将纵轴及横轴取消
axis on	这个命令打开所有的轴标签，核对符号，背景(默认情形)

3.8 练习

3.1 正弦函数的定义为 $\tan\theta=\sin\theta/\cos\theta$ 这个表达能求出角的正弦值，只要 $\cos\theta$ 的值不要太接近 0。假设 θ 用度为单位，编写相应的 **MATLAB** 语句来计算 $\tan\theta$ 的值，只要 $\cos\theta$ 大于等于 10^{-20}，如要小于 10^{-20}，那么打印出错误提示。

3.2 下面的语句用来判断一个人的体温是否处于危险状态（温度用的是华氏计量）。这些语句是否正确？如果不正确，指出错在那里？

```
if temp < 97.5
    disp('Temperature below normal');
elseif temp > 97.5
    disp('Temperature normal');
elseif temp > 99.5
    disp('Temperature slightly high');
elseif temp > 103.0
    disp('Temperature dangerously high');
end
```

3.3 在邮局发一个包裹，不超过两英磅的则收款为 10 美元。超过两英磅每英磅按 3.75 美元来计费，如果包裹的重量超过了 70 英磅，超过了 70 英磅的部分，每英磅的价格为 1.0 美元。如果超过了 100 英磅则拒绝邮递。编写一个程序，输入包裹的重量，输出它的邮费。

3.4 在例 3.3 中我们编写了一个程序用以计算 $f(x,y)$的值。这个函数的定义如下

$$f(x,y)=\begin{cases} x+y & x\ge 0 \text{ and } y\ge 0 \\ x+y^2 & x\ge 0 \text{ and } y<0 \\ x^2+y & x<0 \text{ and } y\ge 0 \\ x^2+y^2 & x<0 \text{ and } y<0 \end{cases}$$

在这里我们要求用 if 的嵌套结构来编写这个程序。

3.5 编写一个程序用以计算以下的函数

$$y(x) = \ln \frac{1}{1-x}$$

x 为自变量，x 的值小于 1。

3.6 编写一个程序允许使用者输入一个字符串，这个字符必须是一个星期中的一天(即"Sunday","Monday","Tuesday"等)，应用 switch 结构把这些字符串转化为相应的数字，以星期天为第一天，以星期六为最后一天。如果输入不是这七个字符串中的一个，那么输出提示信息。

3.7 理想气体定律。理想气体定律定义在例 3.7 中出现。假设 1 mol 的理想气体的体积为 10L，编写程序画出 P－T 图，温度的变化为 250K 到 400K。

3.8 天线增益模式。一个抛物面微波接收天线的接受增益 G 是关于执物面的反射角度 θ 的函数，

$$G(\theta) = |\text{sinc}4\theta| \quad (-\frac{\pi}{2} \le \theta \le \frac{\pi}{2}) \qquad (3.6)$$

其中 $\text{sinc}x=\sin x/x$。用极坐标画出 G 的图象。

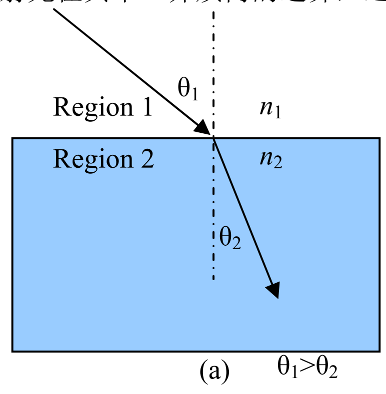
3.9 当光通过不同的介质时，就会发生折射如图(a)(b)。已知满足公式

$$n_1\sin\theta_1=n_2\sin\theta_2 \qquad (3.7)$$

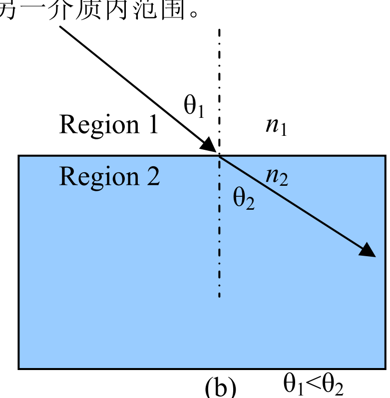
n_1，n_2 分别代表介质 1，2 的折射率，θ_1 代表入射角，θ_2 代表折射角。

$$\theta_2=\sin^{-1}\left(\frac{n_1}{n_2}\sin\theta_1\right) \qquad (3.8)$$

如果 $n_1>n_2$，则入射光将会全部返回 1 介质中，而进入不了 2 介质中。编写一个程序来显示入射光在其中一介质内的边界，还有折射光在另一介质内范围。



(a) $\theta_1>\theta_2$



(b) $\theta_1<\theta_2$

用下面的数据测试你的程序

(a) $n_1 = 1.0, n_2 = 1.7, \theta_1 = 45°$

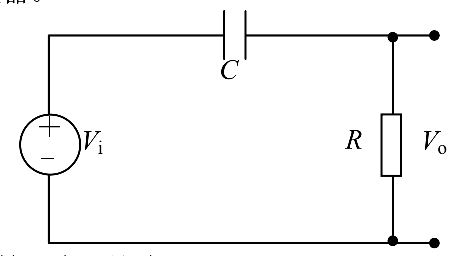
(b) $n_1 = 1.7, n_2 = 1.0, \theta_1 = 45°$

3.10 假设复合函数定义如下

$f(t)=(0.5-0.25i)t-1.0$

计算出相应的函数 f 的幅度与相位。定义域为 $0\leqslant t\leqslant 4$。

3.11 这是一个高通滤波器。



已知高通滤波器的输出输入电压比为

$$\frac{V_o}{V_i} = \frac{j2\pi fRC}{1+j2\pi fRC} \qquad (3.9)$$

计算并画出高通滤波器随频率变化输出电压的幅度和相位。

3.12 阿基米德螺旋。阿基米德螺旋用极坐标可描述为

$$r = k\theta \qquad (3.10)$$

r 是指到原点的距离，θ 为角度，单位为弧度。已知 $0 \leqslant \theta \leqslant 6\pi$，$k$=0.5，画出它的极坐标图。

3.13 内燃机的输出功率。内燃机的输出功率满足以下公式

$$P = \tau_{IND}\omega_m \qquad (3.11)$$

已知 $\omega_m = 188.5(1-e^{-0.2t}) rad/s$，

$\tau_{IND} = 10e^{-0.2t} N \cdot m$，

0<t<10s。

画出两个函数 P 关于 t 的图象，第一个用线性尺度，第二个用对数尺度。

3.14 画轨道。一颗卫星绕地球运行，卫星的轨道是椭圆形的，而地球就处于这个椭圆的某一个焦点上。卫星的轨迹方程满足下式

$$r = \frac{P}{1-\varepsilon\cos\theta} \qquad (3.12)$$

r 与 θ 分加代表卫星距地球的距离和两者形成的交角，P 是体现轨道大小的参数，ε 是来决定轨道形状的参数，ε 为 0 则轨道是圆形的，$0 \leqslant \varepsilon \leqslant 1$ 则说明轨道是椭圆形的。如果 $\varepsilon>1$，则卫星要做离心运动。

已知卫星的 p=1000km，画出卫星的轨迹，已知

(a) ε=0；(b) ε=0.25；(c) ε=0.5

每一颗卫星到地球最近距离是多少?最远距离是多少?比较这三幅图，说出 p 代表意义是什么?

第四章 循环结构

循环(loop)是一种 **MATLAB** 结构，它允许我们多次执行一系列的语句。循环结构有两种基本形式:while 循环和 for 循环。两者之间的最大不同在于代码的重复是如何控制的。在 while 循环中，代码的重复的次数是不能确定的，只要满足用户定义的条件，重复就进行下去。相对地，在 for 循环中，代码的重复次数是确定的，在循环开始之前，我们就知道代码重复的次数了。

4.1 while 循环

只要满足一定的条件，While 循环是一个重复次数不能确定的语句块。它的基本形如下

```
while expression
    ...
    ...   } code block
    ...
end
```

如果 expression 的值非零(true)，程序将执行代码块(code block)，然后返回到 while 语句执行。如果 expression 的值仍然非零，那么程序将会再次执行代码。直到 expression 的值变为 0，这个重复过程结束。当程序执行到 while 语句且 expression 的值为 0 之后，程序将会执行 end 后面的第一个语句。

while 循环的伪代码为

```
while expr
    ...
    ...
    ...
end
```

我们将用 whlie 循环编写一个统计分析的程序。

例 4.1

统计分析在科学工程计算中，跟大量的数据打交道是非常平常的事，这些数据中的每一个数据都是对我们关心的一些特殊值的度量。本课程的第一次测验的成绩就是一个简单的例子。每一个成绩都对某一个学生在本课程中学到多少东西的度量。

许多的时侯，我们并不关心某一个单个数据。我们可以通过总结得到几个重要的数据，以此告诉我们数据的总体情况。例如，一组数据的平均数(数学期望)和标准差。平均数的定义如下:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (4.1)$$

其中 x_i 代表 n 个样本中的第 i 个样本。如果所有的输入数据都可以在一个数组中得到，这些数据的平均数就可以通过公式(4.1)直接计算出来，或应用 **MATLAB** 的内建函数 mean。

标准差的定义如下:

$$s=\sqrt{\frac{N\sum_{i=1}^{N}x_i^2-(\sum_{i=1}^{N}x_i)^2}{N(N-1)}} \tag{4.2}$$

标准差则体现随机变量取值与其期望值的偏差。标准差的值较大，则表明该随机变量的取值与其期望值的偏差较大，反之，则表明此偏差较小。如果所有的输入数据都可以在一个数组中得到，这些数据的平均数就可以通过公式(4.2)直接计算出来，或应用 **MATLAB** 的内建函数 std。本例的目的是要通过公式 4.1，4.2 计算平均数和标准差，向大家介绍 while 循环的应用。我们要执行的算法是读取一个组数据，计算它们的平均数和标准差，最后输出结果。

答案:

程序必须能读取一系列的测量值，并能够计算出这些测量值的数学期望和标准差。在进行计算之前，我们有 while 循环来读取这些测量值。

当所有的测量值输入完毕，我们必须通过一定的方法来告诉程序没有其它的数据输入了。在这里，我们假设所有测量值均为非负数，我们用一个负数来表示数据输入完毕。当一个负数输入时，程序将停止读取输入值，并开始计算这些数据的数学期望和方差。

1.陈述问题因为我们假设所有的输入数据为非负数，则合适地问题陈述为:计算一组测量数的平均数和方差，假设所有测量值为非负数;假设我们事先不知道有多少个测量数。一个负数的输入值将代表测量值输入的结束。

2.定义输入值和输出值这个程序的输入是未知数目的非负数。输出为这些非负数的平均数和标准差。顺便还要打印出输入数据的数据，以便于检测程序的正确性。

3.设计算法这个程序可分为以下三大步:

```
Accumulate the input data
Calculate the mean and standard deviation
Write out the mean, standard deviation, and number of points
```

每一大步的为读取输入值。为达此目的，我们必须提示用户输入合适的值。当数据输入完毕，我们将计算出数据的个数，它们的和与平方和。这些步骤的伪代码如下所示

```
Initialize n, sum_x, and sum_x2 to 0
Prompt user for first number
Read in first x
while x >=0
    n←n+1
    sum_x←sum_x + x
    sum_x2←sum_x2 + x^2
    Prompt user for next number
    Read in next x
end
```

注意我们必须在 while 循环开始之前，我们必须读取第一个值，这样在 while 循环第一次运行中才有了检测值。

下一步，我们要计算出数学期望和标准差。这个步骤的伪代码就是公式（4.1）和(4.2)的 **MATLAB** 版本。

```
x_bar ← sum_x / n
std_dev ← sqrt((n*num_x2 – sum_x^2) / (n*(n-1)))
```

最后我们写出输出结果:

```
Write out the mean value x_bar
Write out the standard deviation std_dev
Write out the number of input data points n
```

4.将伪代码转化为相应的 **MATLAB** 语句最终的 **MATLAB** 程序如下

```
% Script file: stats_1.m
%
% Purpose:
% To calculate mean and the standard deviation of
```

```
% an input data set containing an arbitrary number
% of input values.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/05/97 S. J. Chapman Original code
%
% Define variables:
% n -- The number of input samples
% std_dev -- The standard deviation of the input samples
% sum_x -- The sum of the input values
% sum_x2 -- The sum of the squares of the input values
% x -- An input data value
% xbar -- The average of the input samples
% Initialize sums.
n = 0; sum_x = 0; sum_x2 = 0;
% Read in first value
x = input('Enter first value: ');
% While Loop to read input values.
while x >= 0
    % Accumulate sums.
    n = n + 1;
    sum_x = sum_x + x;
    sum_x2 = sum_x2 + x^2;
    % Read in next value
    x = input('Enter next value: ');
end
% Calculate the mean and standard deviation
x_bar = sum_x / n;
std_dev = sqrt( (n * sum_x2 - sum_x^2) / (n * (n-1)) );
% Tell user.
fprintf('The mean of this data set is: %f\n', x_bar);
fprintf('The standard deviation is: %f\n', std_dev);
fprintf('The number of data points is: %f\n', n);
```

5.检测程序为检测这个程序，我们将手工算出一组简单数据的平均数和标准差，然后与程序产生的结果进行比对。如果我们用三个输入值:3，4 和 5，那么它的平均数和标准差分别为

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{3}\times 12 = 4$$

$$s = \sqrt{\frac{N\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2}{N(N-1)}} = 1$$

我们把这些值输入程序后，产生的结果为

```
>> stats_1
Enter first value: 3
Enter next value: 4
Enter next value: 5
Enter next value: -1
The mean of this data set is: 4.000000
The standard deviation is: 1.000000
The number of data points is: 3.000000
```

这个结果说明了程序的正确性。在这个例子中，我们并没有完全遵循设计过程。这个失误导致这个软件有一个致命的缺陷。你能指出来它吗?

我们的错误就在于我们没有检测程序所有可能的输入类型。请重看一遍程序。如果我们不输入数或者只输入一个数，那么上面的公式就会出现除以 0 的情况。这种除 0 错误将会在导致在命令窗口内出现 divide-by-zero 的警告，导致输出值为无穷大(NaN)。我们需要修改这个程序来解决这个问题，告诉用户这个问题是什么，并在适当的时侯停止。这个程序的修定版本为 stats_2。在运行运算之前，我们必须检查是否有足够多的输入值。如果没有，程序将打印出错误提示信息，并且跳出。你自己检测一下这个版本的程序。

```
% Script file: stats_2.m
%
% Purpose:
% To calculate mean and the standard deviation of
% an input data set containing an arbitrary number
% of input values.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/05/97 S. J. Chapman Original code
% 1. 12/06/97 S. J. Chapman Correct divide-by-0 error if
% 0 or 1 input values given.
%
% Define variables:
% n -- The number of input samples
% std_dev -- The standard deviation of the input samples
% sum_x -- The sum of the input values
% sum_x2 -- The sum of the squares of the input values
% x -- An input data value
% xbar -- The average of the input samples
% Initialize sums.
n = 0; sum_x = 0; sum_x2 = 0;
% Read in first value
x = input('Enter first value: ');
% While Loop to read input values.
while x >= 0
    % Accumulate sums.
    n = n + 1;
    sum_x = sum_x + x;
    sum_x2 = sum_x2 + x^2;
    % Read in next value
    x = input('Enter next value: ');
end
% Check to see if we have enough input data.
if n < 2 % Insufficient information
    disp('At least 2 values must be entered!');
else % There is enough information, so
    % calculate the mean and standard deviation
    x_bar = sum_x / n;
    std_dev = sqrt( (n * sum_x2 - sum_x^2) / (n * (n-1)) );
    % Tell user.
    fprintf('The mean of this data set is: %f\n', x_bar);
    fprintf('The standard deviation is: %f\n', std_dev);
    fprintf('The number of data points is: %f\n', n);
end
```

注意平均数和标准差可以通过 **MATLAB** 的内建函数 mean 和 std 计算得到，输入数据存储在一个向量内，并把向量作为函数的参数。在本章章末的练习中，将会要求你用标准的 **MATLAB** 函数创建一个新的版本程序。

4.2 for 循环

for 循环结构是另一种循环结构，它以指定的数目重复地执行特定的语句块。For 循环的形式如下

```
for index = expr
    Statement 1  ┐
    ...          ├Body
    Statement n  ┘
end
```

其中 index 是循环变量(就是我们所熟知的循环指数),exp 是循环控制表达式。变量 index 读取的是数组 expr 的行数，然后程序执行循环体（loopbody），所以 expr 有多少列，循环体就循环多少次。expr 经常用捷径表达式的]方式，即 first:incr:last。

在 for 和 end 之前的语句我们称之为循环体。在 for 循环运转的过程中，它将被重复的执行。For 循环结构函数如下：

1.在 for 循环开始之时，**MATLAB** 产生了控制表达式

2.第一次进入循环，程序把表达式的第一列赋值于循环变量 index，然后执行循环体内的语句。

3.在循环体的语句被执行后，程序把表达式的下一列赋值于循环变量 index，程序将再一次执行循环体语句。

4.只要在控制表达式中还有剩余的列，步骤 3 将会一遍一遍地重复执行。我们要举大量的例子来说明 for 循环的操作。

第一，考虑下面的例子

```
for ii = 1:10
    Statement 1
    ...
    Statement n
end
```

在这种情况下，控制表达式产生了一个 1 x 10 数组，所以语句 1 到 n 将会被重复执行 10 次。循环系数 ii 在第一次执行的时侯是 1，第二次执行的时侯为 2，依次类推，当最后一次执行时，循环指数为 10。在第十次执行循环体之后，再也没有新的列赋值给控制表达式，程序将会执行 end 语句后面的第一句。注意在循环体在最后一次执行后，循环系数将会一直为 10。

第二，考虑下面的例子。

```
for ii = 1:2:10
    Statement 1
    ...
    Statement n
end
```

在这种情况下，控制表达式产生了一个 1 x 5 数组，所以语句 1 到 n 将会执行 5 次。循环指数 ii 在第一次执行时为 1，第二次执行时为 3，依此类推，最后一次执行时为 9。在第五次执行循环体之后，再也没有新的列赋值给控制表达式，程序将会执行 end 语句后面的第一句。注意在循环体在最后一次执行后，循环系数将会一直为 9。

第三，考虑下面的例子。

```
for ii = [5 9 7]
    Statement 1
    ...
    Statementn
end
```

在这里，控制表达式是一个直接写出的 1 x 3 的数组，所以语句 1 到 n 将会执行 3 次，循环指数 ii 在第一次执行时为 1，第二次执行时为 3，第三次执行时为 7。循环指数在循环结束之后一直为 7。

最后，考虑下面的例子。

```
for ii = [1 2 3; 4 5 6]
    Statement 1
    ...
    Statement n
end
```

在这里，控制表达式是一个直接写出的 2 x 3 的数组，所以语句 1 到 n 将会执行 3 次，循环指数 ii 在第一次执行时为行向量$\begin{bmatrix}1\\4\end{bmatrix}$，第二次执行时为$\begin{bmatrix}4\\5\end{bmatrix}$，第三次执行时为$\begin{bmatrix}6\\7\end{bmatrix}$。这个例子说明循环指数可以为向量。

对应于 for 循环的伪代码为：

```
for index = expression
    Statement 1
    ...
    Statement n
end
```

例 4.2

阶乘（factorial）函数

为了说明 for 循环操作，我们将用 for 循环来计算阶乘函数。阶乘函数的定义如下：

```
N!=1                                    N=0
N!=N * (N-1) * (N-2) * ... * 3 * 2 * 1              N>0
```

计算 N 的阶乘的 **MATLAB** 代码为

```
n_factorial = 1
for ii = 1 : n
    n_factorial = n_factorial * ii;
end
```

假设我们要计算 5 的阶乘。如果 n 为 5，for 循环控制表达式将会产生行向量[1 2 3 4 5]。这种循环将会执行 5 次，ii 值按先后顺序依次为 1，2，3，4，5。n_factorial 最终的计算结果为 1 x 2 x 3 x 4 x 5=120。

例 4.3

计算 the day of year

the day of year 是指这一年已经逝去的天数（包括当天）。在平年中，它的取值范围为 1 到 365，在闰年中，它的取值范围 1 到 366。编写一个 **MATLAB** 程序，输入年，月，日，输入为对应的 the of year。

答案：

为了确定 the day of year，程序需要计算先前月份的天数之后，然后再计算当月已经过去了多少天，在求和的过程中将会用到 for 循环。因为每一个月的天数不尽相同，所以我们要确定每一个月的正确天数。我们用 switch 结构来确定它。

在闰年时，在二月后的某一天的 the day of year 将会比平年时大 1。因为在闰年的二月份多出一个 2 月 29 号。所以为了正确地计算出 the day of year，我们必须确定那一年是闰年。在公历中，闰年是这样规定的

1.能被 400 整除的年为闰年

2.能被 100 整除但不能被 400 整除的年不为闰年

3.能被 4 整除但不能被 100 整除年为闰年

4.其余的年份不为闰年

我们将用到 mod（求余）函数来确定一个数是否能被另一个数整除。如果函数的返回值为 0，则说一个数能被另一个数整除，否则，则不然。

下面是一个用于计算 the day of year 的程序。注意程序如何计算出前面月份总共的天数，如何应用 switch 结构确定每一月的天数。

```
% Script file: doy.m
%
% Purpose:
% This program calculates the day of year corresponding
% to a specified date. It illustrates the use switch
% and for constructs.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/07/98 S. J. Chapman Original code
%
% Define variables:
% day            --Day (dd)
% day_of_year    --Day of year
% ii             --Loop index
% leap_day       --Extra day for leap year
% month          --Month (mm)
% year           --Year(yyyy)
% Get day, month, and year to convert
disp('This program calculates the day of year given the ');
disp('current date.');
month = input('Enter current month (1-12):');
day = input('Enter current day(1-31):');
year = input('Enter current year(yyyy): ');
% Check for leap year, and add extra day if necessary
if mod(year,400) == 0
leap_day = 1; % Years divisible by 400 are leap years
elseif mod(year,100) == 0
leap_day = 0; % Other centuries are not leap years
elseif mod(year,4) == 0
leap_day = 1; % Otherwise every 4th year is a leap year
else
leap_day = 0; % Other years are not leap years
end
% Calculate day of year by adding current day to the
% days in previous months.
day_of_year = day;
for ii = 1:month - 1
% Add days in months from January to last month
switch (ii)
case {1,3,5,7,8,10,12},
day_of_year = day_of_year + 31;
case {4,6,9,11},
day_of_year = day_of_year + 30;
case 2,
day_of_year = day_of_year + 28 + leap_day;
end
end
% Tell user
fprintf('The date %2d/%2d/%4d is day of year %d.\n', ...
```

```
            month, day, year, day_of_year);
```

我们用下面已知的结果来检测这个程序。

1.1999 年不是闰年。它的 1 月 1 号对应的 day of year 是 1，12 月 31 号必定对应的是 365。

2.2000 年是一个闰年。它的 1 月 1 号对应的 day of year 是 1，12 月 31 号必定对应的是 366。

3.2001 年不是闰年。它的 1 月 1 号对应的 day of year 是 30。这个程序 5 次运行后的结果分别为

```
>> doy
This program calculates the day of year given the
current date.
Enter current month (1-12):1
Enter current day(1-31):1
Enter current year(yyyy): 1999

The date   1/ 1/1999 is day of year 1.

>> doy
This program calculates the day of year given the
current date.
Enter current month (1-12):12
Enter current day(1-31):31
Enter current year(yyyy): 1999
The date 12/31/1999 is day of year 365.

>> doy
This program calculates the day of year given the
current date.
Enter current month (1-12):1
Enter current day(1-31):1
Enter current year(yyyy): 2000

The date   1/ 1/2000 is day of year 1.

>> doy
This program calculates the day of year given the
current date.
Enter current month (1-12):12
Enter current day(1-31):31
Enter current year(yyyy): 2000

The date 12/31/2000 is day of year 366.

>> doy
This program calculates the day of year given the
current date.
Enter current month (1-12):3
Enter current day(1-31):1
Enter current year(yyyy): 2001

The date   3/ 1/2001 is day of year 60.
```

通过 5 次不同情况的检测，这个程序给出了正确的结果。

例 4.4

统计分析

执行如下算法：

输入一系列的测量数，计算它们的平均数和标准差。这些数可以是正数，负数或 0。

答案：

这个程序必须能够读取大量数据，并能够计算出这些测量值的平均数和标准差。这些测量值可以是正数，负数或 0。

因为我们再也不能用一个数来表示数据中止的标识了，我们要求用户给出输入值的个数，然后用 for 循环读取所有数值。

下面的就是这个修定版本的程序。它允许各种输入值，请你自己验证下面 5 个输入值的平均数和标准差：3，-1，0，1，-2。

```
% Script file: stats_3.m
%
% Purpose:
% To calculate mean and the standard deviation of
% an input data set, where each input value can be
% positive, negative, or zero.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/08/97 S. J. Chapman Original code
%
% Define variables:
% ii Loop index
% n The number of input samples
% std_dev The standard deviation of the input samples
% sum_x The sum of the input values
% sum_x2 The sum of the squares of the input values
% x An input data value
% xbar The average of the input samples
% Initialize sums.
sum_x = 0; sum_x2 = 0;
% Get the number of points to input.
n = input('Enter number of points: ');
% Check to see if we have enough input data.
if n < 2 % Insufficient data
    disp ('At least 2 values must be entered.');
else % we will have enough data, so let's get it.
    % Loop to read input values.
    for ii = 1:n
        % Read in next value
        x = input('Enter value: ');
        % Accumulate sums.
        sum_x = sum_x + x;
        sum_x2 = sum_x2 + x^2;
    end
    % Now calculate statistics.
    x_bar = sum_x / n;
    std_dev = sqrt((n * sum_x2 - sum_x^2) / (n * (n - 1)));
    % Tell user.
    fprintf('The mean of this data set is: %f\n', x_bar);
    fprintf('The standard deviation is: %f\n', std_dev);
```

```
        fprintf('The number of data points is: %f\n', n);
end
```

4.2.1 运算的细节

既然我们已经看了许多 for 循环的例子。在用 for 循环时，我们必须检查许多重要的细节。

1.没有必要缩进 for 循环的循环体。即使所有语句都左对齐，**MATLAB** 程序也会识别出这个循环。但缩进循环体能增强代码的可读性，所以建议大家缩进循环体。

好的编程习惯
对于 for 循环体总是要缩进两个或更多空格，以增强程序的可读性。

2.在 for 循环中，我们不能随意修改循环指数。循环指数常被用作计算器，

如果修改了它们将会导致一些奇怪而难以发现的错误。下面的一个例子将初始化一个函数的数组。但是语句“ii=5”的突然出现，导致只有 a(5)得到了初始化，它得到了本应赋给 a(1)，a(2)等等的值。

```
for ii = 1:10
    ...
    ii = 5 ; %Error!
    ...
    a(ii) = <calculation>
end
```

好的编程习惯
在循环体中绝不修改循环指数的值。

3.我们在第二章已经学过，用赋值的方法可以扩展一个已知的数组。例如，语句

```
arr = 1:4;
```

定义了一个数组[1 2 3 4]。如果执行语句

```
arr(8) = 6;
```

将会产生一个八元素数组[1 2 3 4 0 0 0 6]。不幸的是，每一次扩展数组，都要经过以下步骤:第一步，创建一个新数组。第二步，把旧数组的元素复制到新数组当中。第三步，把扩展的元素写入新数组。第四步，删除旧数组。对于大数组来说这些步骤是相当耗时的。

当一个 for 循环中存储了一个预先未定义的数组，在第一次循环执行的时侯，循环结构迫使 **MATLAB** 重复执行以上步骤。从另一方面说，如果在循环开始之前数组预先分配了数组的大小，那么复制就不需要了，执行代码的速度也将加快。下面代码片段向大家展示了在循环开始之前如何预先分配数组。

好的编程习惯
在循环执行开始之前，总是要预先分配一个数组，这样能大大增加循环运行的速度。

4.用 for 循环和向量计算是非常常见的。例如，下面的代码片段用 for 循环计算 1 到 100 之间的所有整数的平方，平方根，立方根。

```
for ii = 1:100
    square(ii) = ii ^2;
    square_root(ii) = ii ^ (1/2);
    cube_root(ii) = ii ^ (1/3);
end
```

下面一个代码片段是用向量计算上面的问题。

```
    ii = 1:100;
```

```
square = ii .^2;
square_root = ii .^ (1/2);
cube_root(ii) = ii .^ (1/3);
```

尽管两种算法得到了相同的结果，但两者并不同等价。因为 for 循环算法比向量算法慢 15 倍还多。这是由于 **MATLAB** 通过每一次循环时，每行都要翻译执行一次。也相当于 **MATLAB** 翻译执行了 300 行代码。相反，如果用向量算法，**MATLAB** 只需要翻译执行 4 行代码。所以用向量语句它的执行速度非常快。

向量算法的缺点是需要很大的内存，因为一些间接的数组需要创建。这经常是一小点损失，所以要比 for 循环算法好的多。

在 **MATLAB** 中，用向量算法代替循环的算法的过程称之为向量化(vectorization)。向量化能够改进许多的 **MATLAB** 程序。

好的编程习惯

那种既可以用向量可以解决的问题，也可以用循环解决的问题，最好用向量解决，这是因为向量执行的速度快。

例 4.5

比较向量算法和循环为了比较循环和向量算法执行若无事所用的时间，用两种方法编程并测试三个运算所花的时间。

1.用 for 循环计算 1 到 10000 的之间每一个整数的平方，而事先不初始化平方数组。

2.用 for 循环计算 1 到 10000 的之间每一个整数的平方，而事先初始化平方数组。

3.用向量算法计算计算 1 到 10000 的之间每一个整数的平方。

答案:

这个程序必须用上面提供的三种方示计算出 1 到 10000 之间的每一个整数的平方，并测试每一个种算法的时间。测试时间要用到 **MATLAB** 函数 tic 和 toc。tic 函数复位内建计时器，而 toc 函数则从最后一次调用 tic 以秒开始计时。

因为在许多的计算机中它的时间钟是相当粗略的，所以有必要多运行几次以获得相应的平均数。

下面就是用三种方法编出的 **MATLAB** 程序。

```
% Script file: timings.m
%
% Purpose:
% This program calculates the time required to
% calculate the squares of all integers from 1 to
% 10,000 in three different ways:
% 1. Using a for loop with an uninitialized output
% array.
% 2. Using a for loop with an preallocated output
% array.
% 3. Using vectors.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/08/97 S. J. Chapman Original code
%
% Define variables:
% ii, jj Loop index
% average1 Average time for calculation 1
% average2 Average time for calculation 2
```

```
% average3 Average time for calculation 3
% maxcount Number of times to loop calculation
% square Array of squares
% leap_day Extra day for leap year
% month Month(mm)
% year Year(yyyy)
% Perform calculation with an uninitialized array
% "square". This calculation is done only once
% because it is so slow.
maxcount = 1; % One repetition
tic; % Start timer
for jj = 1:maxcount
    clear square % Clear output array
    for ii = 1:10000
        square(ii) = ii^2; % Calculate square
    end
end
average1 = (toc)/maxcount; % Calculate average time
% Perform calculation with a preallocated array
% "square". This calculation is averaged over 10
% loops.
maxcount = 10; % One repetition
tic; % Start timer
for jj = 1:maxcount
    clear square % Clear output array
    square = zeros(1,10000); % Preinitialize array
    for ii = 1:10000
        square(ii) = ii^2; % Calculate square
    end
end
average2 = (toc)/maxcount; % Calculate average time
% Perform calculation with vectors. This calculation
% averaged over 100 executions.
maxcount = 100; % One repetition
tic; % Start timer
for jj = 1:maxcount
    clear square % Clear output array
    ii = 1:10000; % Set up vector
    square = ii.^2; % Calculate square
end
average3 = (toc)/maxcount; % Calculate average time
% Display results
fprintf('Loop / uninitialized array = %8.4f\n', average1);
fprintf('Loop / initialized array = %8.4f\n', average2);
fprintf('Vectorized = %8.4f\n', average3);
```

4.2.2 break 和 continue 语句

有两个附加语句可以控制 while 和 for 循环:break 和 continue 语句。break 语句可以中止循环的执行和跳到 end 后面的第一句执行，而 continue 只中止本次循环，然后返回循环的顶部。如果 break 语句在循环体中执行，那么体的执行中止，然后执行循环后的第一个可执行性语句。用在 for 循环中的 break 语句的例子如下:

程序执行的结果为:

```
%test_break.m
for ii = 1:5;
    if ii == 3;
        break;
    end
    fprintf('ii = %d \n', ii);
end
disp('End of loop!');
>> test_break
ii = 1
ii = 2
End of loop!
```

注意 break 语句在 ii 为 3 时执行，然后执行 disp('End of loop!');语句而不执行 fprintf('ii = %d \n', ii);语句。

continue 语句只中止本次循环，然后返回循环的顶部。在 for 循环中的控制变量将会更新到下一个值，循环将会继续进行。下面是一个在 for 循环中的 continue 的例子。

```
%test_continue.m
for ii = 1:5;
    if ii == 3;
        continue;
    end
    fprintf('ii = %d \n', ii);
end
disp('End of loop!');
```

程序运行的结果为;

```
>> test_continue
ii = 1
ii = 2
ii = 4
ii = 5
End of loop!
```

注意 continue 语句在 ii 为 3 时执行，然后程序返回循环的顶部而不执行 fprintf 语句。break 和 continue 语句可用在 while 循环和 for 循环中。

4.2.3 循环嵌套

一个循环完全出现在另一个循环当中，这种情况经常发生。如果一个循环完全出现在另一个循环当中，我们称这两个循环为**带嵌套的循环**。下面的例子用两重 for 循环嵌套来计算并写出结果。

```
for ii = 1:3
    for jj = 1:3
        product = ii * jj;
        fprintf('%d * %d = %d \n',ii,jj,product);
    end
end
```

在这个例子中，外部的 for 循环将把 1 赋值于循环指数 ii，然后执行内部 for 循环。内部循环的循环体将被执行 3 次，它的循环指数 ii 将会先后被赋值为 1，2，3。当完全执行完内部的循环后，外部的 for 循环将会把 2 赋值于循环指数 ii，然后内部的 for 循环将会再次执行。直到外部 for 循环执行 3 次，这个重复过程结束。产生的结果为

```
1 * 1 = 1
1 * 2 = 2
```

```
1 * 3 = 3
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
```

注意外部 for 循环指数变量增加之前，内部 for 循环要完全执行完。

当 **MATLAB** 遇到一个 end 语句，它将与最内部的开放结构联合。所以第一个 end 语句与语句“for jj = 1:3”，第二个 end 语句与语句“for ii = 1:3”联合。如果在循环嵌套中一个 end 语句突然被删除，将会产生许多难以发现的错误。

如果 for 循环是嵌套的，那么它们必须含有独立的循环变量。如果它们含有相同的循环变量，那么内部循环将改变外部循环指数的值。

如果 break 或 continue 语句出现在循环嵌套的内部，那么 break 语句将会在包含它的最内部的循环起作用。

```
for ii = 1:3
    for jj = 1:3
        if jj ==3;
            break;
        end
        product = ii * jj;
        fprintf('%d * %d = %d \n',ii,jj,product);
    end
    fprintf('End of inner loop\n');
end
fprintf('End of outer loop\n');
```

如果内部循环指数 jj 为 3，那么 break 语句开始执行，这将导致程序跳出内部循环。程序将会打印出”End of inner loop”，外部循环指数将会增加 1，内部循环的执行重新开始。产生的输出值为:

```
1 * 1 = 1
1 * 2 = 2
End of inner loop
2 * 1 = 2
2 * 2 = 4
End of inner loop
3 * 1 = 3
3 * 2 = 6
End of inner loop
End of outer loop
```

4.3 逻辑数组与向量化

在第二章中，我们提出 **MATLAB** 有两个基本类型的数据类型:数字型与字符型。数字型数据包括数字，字符型数据包含字符。除这两个数据类型之外，还有第三类数据类:逻辑型。

“逻辑”数据类型在 **MATLAB** 中并不真实存在。其实，它是带特定逻辑属性标准数字型数据类型。逻辑型数组通过所有的关系运算符和逻辑运算符创建。它们区别于数字型的是在调用 whos 命令时，(logical)会出现在类型的后面。

例如，考虑下面的语句

```
a = [1 2 3; 4 5 6; 7 8 9];
b = a > 5;
```

这些语句将会产生两个数组 a 和 b。a 将会产生一个数组$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$，b 将会产生一个特殊的含有逻辑属性$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$。当调用 whos 命令时，结果如下。注意 b 后面的(logical)修饰符。

```
>> whos
  Name        Size                    Bytes  Class

  a           3x3                        72  double array
  b           3x3                         9  logical array

Grand total is 18 elements using 81 bytes
```

我们还可以用 logical 函数给一个数组加上一个逻辑属性。例如，语句 c=logical(a)，将会把 a 值赋于 c，从而使 c 带有一定的逻辑性:

一个数组的逻辑属性可以通任何的数学运算去除。例如，如果我们在 c 数组加 0，数组的值不会改变，而它的逻辑属性将会消失

```
>> c=b+0

c =

     0     0     0
     0     0     1
     1     1     1

>> whos
  Name        Size                    Bytes  Class

  a           3x3                        72  double array
  b           3x3                         9  logical array
  c           3x3                        72  double array

Grand total is 27 elements using 153 bytes
```

4.3.1 逻辑数组的重要性

逻辑数组有一个重要的属性——它在算术运算中能提供一个屏蔽(mask)。屏蔽(mask)是指一个数组，它从另一个数组选择所需的元素参与运算。指定的运算只在选择的元素上执行，而不执行原有的元素。

例如，假设数组 a 和 b 的定义如上节所示。那么语句 a(b)=sqrt(a(b))会计算 a 中相应的元素的平方根，相应的元素是指与 b 数组中的非零元素相对应的数组 a 中的元素。其他元素保持不变。

```
>> a(b)=sqrt(a(b))

a =

    1.0000    2.0000    3.0000
    4.0000    5.0000    2.4495
    2.6458    2.8284    3.0000
```

对于一个数组的子集快速而简单，而不用循环和选择结构。

下面的语句，是用循环结构和选择结构计算上述问题。

```
for ii = 1:size(a,1)
    for jj = 1:size(a,2)
        if a(ii,jj) > 5
            a(ii,jj)=sqrt(a(ii,jj));
        end
    end
end

b = a > 5;
a(b) = sqrt(a(b));
```

例 4.6

用逻辑数数组进行屏蔽运算为了比较循环结构，选择结构与应用逻辑数组运算的快慢，我们进行下面两个计算，并对它进行计时。

1.创建一个含 10000 个元素的数组，其值依次为 1 到 10000 之间的整数。用 for 循环和 if 结构计算大于 5000 的元素的平方根。

2.创建一个含 10000 个元素的数组，其值依次为 1 到 10000 之间的整数。用逻辑数组计算大于 5000 的元素的平方根。

答案:

这个程序必须创建一个含 10000 个元素的数组，其值依次为 1 到 10000 之间的整数。用两种不同的方法计算出大于 5000 的元素的平方根。

比较两种方法运行速度的 **MATLAB** 程序如下所示:

```
% Script file:logical1.m
%
% Purpose:
%     This program calculates the time required to
%     calculate the square roots of all elements in
%     array a whose value exceeds 5000. This is done
%     in two differents ways:
%     1.Using a for loop and if construct.
%     2.Using a logical array.
%
% Record of revisions:
%     Date          Programmer          Description of change
%     ====          ================    ==============================
% 06/01/02       S. J. Chapman          Original code
%
% Define variables:
%     a                  --Array of input values
%     b                  --Logical array to serve as a mask
%     ii,jj              --Loop index
%     average1           --Average time for calculation 1
%     average2           --Average time for calculation 2
%     maxcount           --Number of times to loop calculation
%     month              --Month (mm)
%     year               --Year (yyyy)
%
% Perform calculation using loops and branches
maxcount = 1;                  % One repetition
tic;                           % Start timer
```

```
for jj = 1:maxcount
    a = 1:10000;          %Declare array a
    for ii = 1:10000
        if a(ii) > 5000
            a(ii) = sqrt(a(ii));
        end
    end
end
average1 = (toc)/maxcount;%Calculate average time
%
% Perform calculation using logical arrays.
maxcount = 10;        %One repetition
tic;                  %Start timer
for jj = 1:maxcount
    a = 1:10000;      %Declare array a
    b = a > 5000;     %Create mask
    a(b) = sqrt(a(b));    %Take square root
end
average2 = (toc)/maxcount;   %Calculate average time
%
% Display result
fprintf('Loop /if approach = %8.4f\n',average1);
fprintf('Logical array approach = %8.4f\n',average2);
```

这个程序在 cpu 为奔腾 III(主频为 733mhz)的计算机运行得到结果如下

```
>> logical1
Loop /if approach =      0.1200
Logical array approach =      0.0060
```

正如我们看到的，用逻辑数组方法速度是另一种方法的 20 倍。

好的编程习惯

如果用可能的话，可用逻辑函数选择数组中的元素。如果逻辑数组进行运算，要比循环快得多。

4.3.2 用 if/else 结构和逻辑数组创建等式

逻辑数组经常被用来替代 for 循环中的 if/else 结构。正如我们在上节所看到的，把逻辑运算当作一个屏蔽来选择数组中的某些元素进行运算。如果你要利用那些没有被选择到的元素进行运算，只需要在逻辑屏蔽上加一个非运算符(-)。例如，假设我们要计算一个二维数组中所有的大于 5 的元素的平方根，然后其余的数的平方。利用循环和选择结构的代码如下:

```
for ii = 1:size(a,1)
    for jj = 1:size(a,2)
        if a(ii,jj) > 5
            a(ii,jj) = sqrt(a(ii,jj));
        else
            a(ii,jj) = a(ii,jj)^2;
        end
    end
end
```

用逻辑数组运算的代码如下:

```
b = a > 5
a(b) = sqrt(a(b));
a(~b) = a(~b) .^2;
```

显然用逻辑数组的方法运算速度要快得多。

测试 4.1

本测试提供了一个快速的检查方式，看你是否掌握了 4.1 到 4.3 的基本内容。如果你对本测试有疑问，你可以重读 4.1 到 4.3，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

检测下列 for 循环，说出它们的循环次数:

```
1. for index = 7:10
2. for jj = 7:-1:10
3. for index = 1:10:10
4. for ii = -10:3:-7
5. for kk = [0 5; 3 3]
```

检测下列循环，确定循环指数 ires 的最终值。

```
6.  ires = 0;
     for index = 1:10;
          ires = ires + 1;
     end

7.  ires = 0;
     for index = 1:10;
          ires = ires + index;
     end

8.  ires = 0;
     for index1 = 1:10;
          for index2 = index1:10
               if index2 == 6
                    break;
               end
               ires = ires + 1;
          end
     end

9.  ires = 0;
     for index1 = 1:10;
          for index2 = index1:10
               if index2 == 6
                    continue;
               end
               ires = ires + 1;
          end
     end
```

10. 编写一个 **MATLAB** 语句，计算下列的函数值

定义域为$-6\pi<t<6\pi$，每隔 π 取一次值。用两种方法进行运算，一次用循环和选择语句，

$$f(t)=\begin{cases}\sin t & \text{for all t where sint} > 0 \\ 0 & \text{elsewhrer}\end{cases}$$

另一次用逻辑数组。

4.4 附加例子

例 4.7

用最小二乘法画噪声数据的近似曲线

下落物体将会作匀加速度运动，它的速度符合下面的公式

$$v(t) = at + v_0 \qquad (4.3)$$

$v(t)$代表物体在 t 时刻的速度。加速度为 g，初速度 v_0 为 0。这个公式出现在基础物理学中，我们大家都非常的熟悉。如果我们要画出下落物体的速度时间图象，我们得到的(v，t)测量值应当在同一条直线上。但是，学习物理的同学都知道，在实验室得到的测量值不一定是直线。为什么会这样呢?因为所有的测量都有误差。在所有测量值中都有一定的噪声。

在工程和科研方面，有许多像这个例子一样带有噪声，而我们希望得到最符合的结果。这个问题叫做线性待定问题。给出一系列带噪声的测量值(x，y)，它遵循一条直线，如何确定“最符合”这条直线的解析式呢。

如果我们确定了待定系数 m 和 b，那么我们就确定了解析式 4.4。

$$y=mx+b \qquad (4.4)$$

确定待定系数 m 和 b 的标准方法为最小二乘法。之所以称为最小二乘法，是因为根据偏差的平方和为最小的条件来选择常数 m 和 b 的。公式如下:

$$m = \frac{(\sum xy) - (\sum x)\bar{y}}{(\sum x^2) - (\sum x)\bar{x}} \qquad (4.5)$$

$$b = \bar{y} - m\bar{x} \qquad (4.6)$$

其中，$\sum x$ 代表所有测量值 x 之和，$\sum y$ 代表所有测量值 y 之和，$\sum xy$ 代表所有对应的 x 与 y 的乘积之和，$\bar{x}$ 代表测值量 x 的数学期望。$\bar{y}$ 代表测值量 y 的数学期望。已知有一系列含有噪声的数据(x，y)，编写程序用最小二乘法计算出 m 和 b。数据要求从键盘输入，画出每一个数据点还有画出最适合的直线。

答案:

1. 陈述问题

已知有一系列含有噪声的数据(x，y)用最小二乘法计算 m 和 b。数据要求从键盘输入，画出每一个数据点还有画出最适合的直线。

2. 定义输入输出值

这个程序所需的输入值为点的个数，以及点的坐标。输出是用最小二乘法得到的斜率以及 y 上的截距。

3. 设计算法

这个问题被分解为 6 个大步骤:

```
Get the number of input data points
Read the input data values
Calculate the required statistics
Calculate the slop and intercept
Write out the slop and intercept
Plot the input points and the fitted line
```

第一大步是读取输入量的个数，所以我们要用到 input 函数。下一步我们要在 for 循环中使用 input 函数读取输入量(x，y)。每一个输入值将会产生一个数组([x，y])，然后这个函数将会返回这个数组到调用程序。注意在这里应用 for 循环是合适的，因为我们事先知道循环要执行多少次。

上述步骤的伪代码如下:

```
Print message describing purpose of the program
```

```
n_points ← input('Enter number of [x y] pairs:');
for ii = 1:n_points
    temp ← ('Enter [x y] pairs:');
    x(ii) ← temp(1);
    y(ii) ← temp(2);
end
```

下一步，我们必须计算出相关的统计量。这些统计量是$\sum x$，$\sum y$，$\sum xy$，$\sum x^2$。伪代码如下

```
Clear the vairables sum_x, sum_y, sum_x2, and sum_xy
for ii = 1:n_points
    sum_x ← sum_x + x(ii)
    sum_y ← sum_y + y(ii)
    sum_x2 ← sum_x2 + x(ii)^2
    sum_xy ← sum_xy + x(ii) * y(ii)
end
```

下一步我们必须计算出斜率 m 和 y 轴上的截距 b。这一步的伪代码就是公式(4.4)和(4.5)。

```
x_bar ← sum_x / n_points
y_bar ← sum_y / n_points
slope ← (sum_xy-sum_x * y_bar)/( sum_x2 –sum_x * x_bar)
y_int ← y_bar – slope * x_bar
```

最后，我们必须写出和画出相应的结果。输入的坐标点要用圆点画出，不用连接线而用最小二乘法得到解析式对应的直线用 2pixel 的实直线画出。在此之前我们要用到 hold on 命令。画完直线之后，调用 hold off 命令。我们在图象中将会添加相应的标题，以及相应的图例。

4. 转化为 **MATLAB** 语句

```
% Script file: lsqfit.m
% Purpose:
% To perform a leastsquares fit of an input data set
% to a straight line, and print out the resulting slope
% and intercept values. The input data for this fit
% comes from a userspecified input data file.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 01/03/99 S. J. Chapman Original code
%
% Define variables:
% ii Loop index
% n_points Number in input [x y] points
% slope Slope of the line
% sum_x Sum of all input x values
% sum_x2 Sum of all input x values squared
% sum_xy Sum of all input x*y yalues
% sum_y Sum of all input y values
% temp Variable to read user input
% x Array of x values
% x_bar Average x value
% y Array of y values
% y_bar Average y value
% y_int yaxis intercept of the line
disp('This program performs a leastsquares fit of an ');
disp('input data set to a straight line.');
n_points = input('Enter the number of input [x y] points: ');
```

```
% Read the input data
for ii = 1:n_points
temp = input('Enter [x y] pair: ');
x(ii) = temp(1);
y(ii) = temp(2);
end
% Accumulate statistics
sum_x = 0;
sum_y = 0;
sum_x2 = 0;
sum_xy = 0;
for ii = 1:n_points
    sum_x = sum_x + x(ii);
    sum_y = sum_y + y(ii);
    sum_x2 = sum_x2 + x(ii)^2;
    sum_xy = sum_xy + x(ii) * y(ii);
end
% Now calculate the slope and intercept.
x_bar = sum_x / n_points;
y_bar = sum_y / n_points;
slope = (sum_xy - sum_x * y_bar) / ( sum_x2 - sum_x * x_bar);
y_int = y_bar - slope * x_bar;
% Tell user.
disp('Regression coefficients for the leastsquares line:');
fprintf(' Slope (m) = %8.3f\n', slope);
fprintf(' Intercept (b) = %8.3f\n', y_int);
fprintf(' No of points = %8d\n', n_points);
% Plot the data points as blue circles with no
% connecting lines.
plot(x,y,'bo');
hold on;
% Create the fitted line
xmin = min(x);
xmax = max(x);
ymin = slope * xmin + y_int;
ymax = slope * xmax + y_int;
% Plot a solid red line with no markers
plot([xmin xmax],[ymin ymax],'r','LineWidth',2);
hold off;
% Add a title and legend
title ('\bfLeastSquaresFit');
xlabel('\bf\itx');
ylabel('\bf\ity');
legend('Input data','Fitted line');
grid on
```

5. 检测程序为了检测这个程序，我们将采用一些简单的数据，如果输入数据所对应的点都在同一条

直线，那么产生的斜率和截距必定是那条直线的斜率和截距。这组数据为

```
[1.1 1.1]
[2.2 2.2]
[3.3 3.3]
[4.4 4.4]
[5.5 5.5]
[6.6 6.6]
[7.7 7.7]
```

它的斜率和截距分别为 1.0 和 0.0。我们将用这些值来运行这个程序，结果如下:

```
>> lsqfit
This program performs a leastsquares fit of an
input data set to a straight line.
Enter the number of input [x y] points: 7
Enter [x y] pair: [1.1 1.1]
Enter [x y] pair: [2.2 2.2]
Enter [x y] pair: [3.3 3.3]
Enter [x y] pair: [4.4 4.4]
Enter [x y] pair: [5.5 5.5]
Enter [x y] pair: [6.6 6.6]
Enter [x y] pair: [7.7 7.7]
Regression coefficients for the leastsquares line:
 Slope (m) =       1.000
 Intercept (b) =       0.000
 No of points =              7
```

图象如图 4.1(a)

我们将在这些值上加入一些噪声，如下所示

```
 [1.1 1.01]
 [2.2 2.30]
 [3.3 3.05]
 [4.4 4.28]
 [5.5 5.75]
 [6.6 6.48]
 [7.7 7.84]
```

再次运行程序，所得的结果如下:

```
>> lsqfit
This program performs a leastsquares fit of an
input data set to a straight line.
Enter the number of input [x y] points: 7
Enter [x y] pair: [1.1 1.01]
Enter [x y] pair: [2.2 2.30]
Enter [x y] pair: [3.3 3.05]
Enter [x y] pair: [4.4 4.28]
Enter [x y] pair: [5.5 5.75]
Enter [x y] pair: [6.6 6.48]
Enter [x y] pair: [7.7 7.84]
Regression coefficients for the leastsquares line:
 Slope (m) =       1.024
 Intercept (b) =     -0.120
 No of points =              7
```

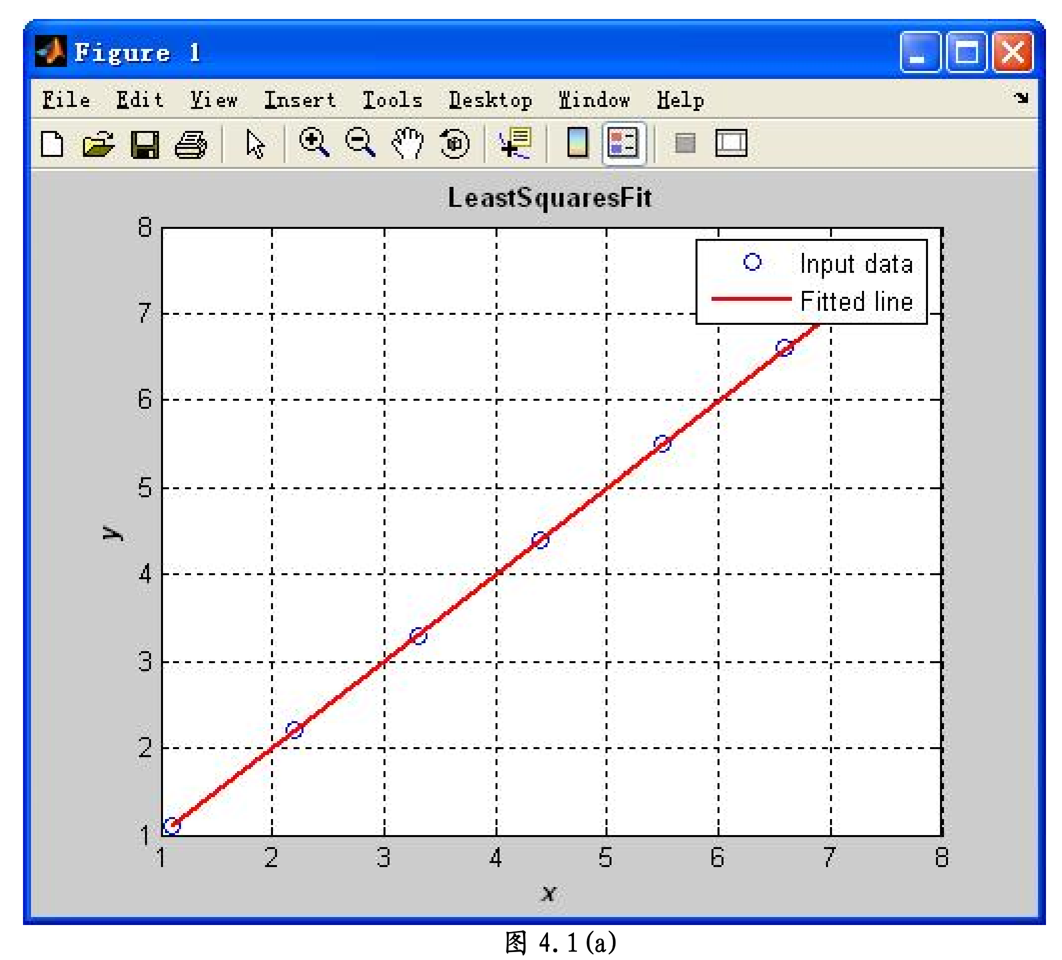


图 4.1 (a)

我们用手动计算很容易就能得到上面两个程序的正确结果。第二个程序图象如图 4.2(b)所示。

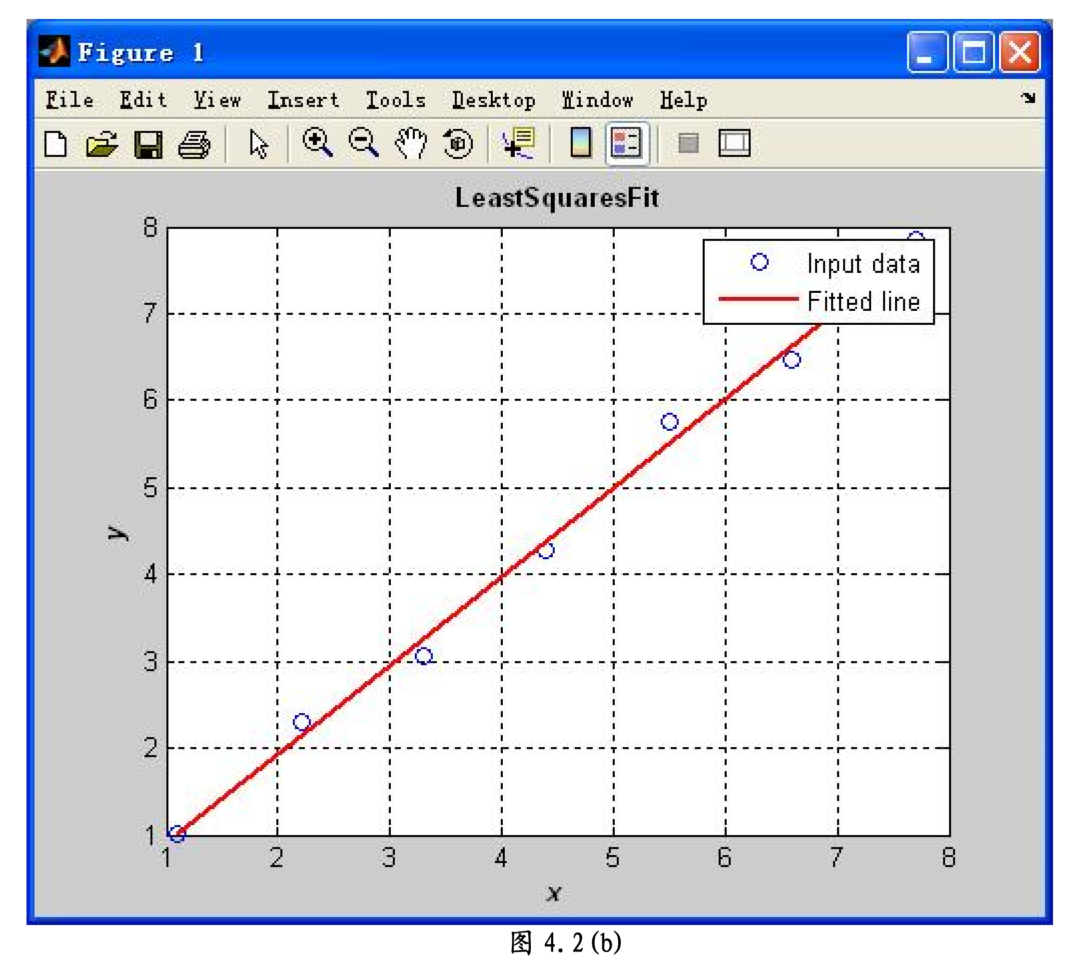


图 4. 2 (b)

这个例子用到了第三章中许多画图的例子。用 hold 命令在同一坐标下，画出了多个图象。用 LineWidth 属性来改变直线的宽度。用转义序列来设标题字体。

例 4.8

小球的轨迹如果我们假设处于真空中，且忽略地球的曲率。在地球任意一点向空中抛出一小球将会产生类似于图 4.2（a）所示的曲线。球在时刻 t 的高度将会遵守公式(4.7)。

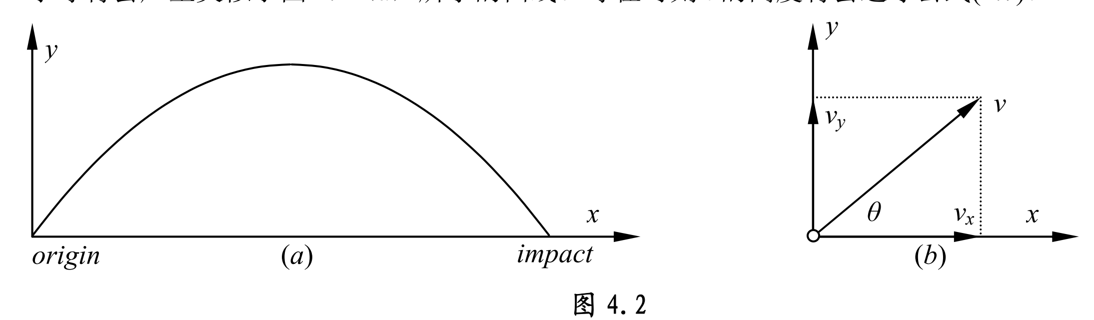


图 4. 2

(*a*) 如果物体上抛，将会产生一个抛特线轨迹。(*b*) 速度可以分解为水平速度和竖直速度，水平速度和合速度之间的夹角为 *θ*。

$$y(t) = y_0 + v_{y0}t + \frac{1}{2}gt^2 \qquad (4.7)$$

其中 y_0 是初始高度，v_{y0} 代表初速度，g 代表重力加速度。水平位移的计算公式为

$$x(t)=x_0+v_{x0}t \qquad (4.8)$$

x_0代表初始位移，v_{x0}代表这个物体的水平初速度水平初速度，竖直初速度与合初速度之间的关系为

$$v_{x0}=v_0\cos\theta \quad (4.9)$$

$$v_{y0}=v_0\sin\theta \quad (4.10)$$

假设一个小球的初始位置为(x_0，y_0)为(0，0)，初速度为 20m/s，水平速度和合速度之间的夹角为 θ 度，编写一个程序，画出这个小球的轨迹，并计算小球再次落地与初始位置之间的距离。这程序应当能画出多个抛物线，θ 的取值从 5 到 85 度，每隔 10 度取一次，计算水平位移，θ 的取值从 0 到 90 度，每隔 1 度取一次。最终应当确定那一个 θ 值使得水平位移最大。还有打印不同的抛物线时要用不同的颜色。

答案:

为了解决问题，我们必须知道这个小球落地的时间。然后我们 4.7 到 4.10 计算出小球的位置。如果多次取不同的值，我们将画出小球的轨迹。

小球落地的时间 T，可以通过公式 4.7 得到。满足 $y(t)$=0。

$$y(t)=y_0+v_{yo}t+\frac{1}{2}gt^2 \quad (4.7)$$

$$0=0+v_{yo}t+\frac{1}{2}gt^2$$

$$0=(v_{yo}+\frac{1}{2}gt)t$$

t=0 舍去，故得到下面的值。

$$t_2=-\frac{2v_{y0}}{g} \quad (4.11)$$

从问题中我们可知，小球的初始位置为(x_0，y_0)为(0，0)，初速度为 20m/s，计算水平位移，θ 的取值从 0 到 90 度，每隔 1 度取一次。最后我们从基础物理学的课本可知，地球的重力加速度为 9.81m/s^2。

1. 陈述问题

当一个小球以初始角度 θ，初速度 v_0 抛出计算这个小球的落地位移。θ 的取值从 0 到 90 度，每隔 1 度取一次。确定那一个角度的水平位移最大。用出一系列的抛物线，这时 θ 的取值从 5 到 85 度，每隔 10 度取一次。用不同的颜色且粗一点的线画出落地位移最大的抛物线。

2. 定义输入量和输出量根据问题的定义，我们知道不需要输入量。输出为水平位移为最大时的 θ 角和指定抛物线的图象。

3. 设计算法问题分为 5 大步

```
Calculate the range of the ball for θ between 0 and 90°
Write a table of ranges
Determine the maximum range and write it out
Plot the trajectories for θ between 5 and 85°
Plot the maximum-range trajectory
```

因为我们精确地知道循环重复的次数，所以在这里用 for 循环是合适的。我们现在开始定义每一个大步骤的伪代码。为了计算每一个角度小球的落地位移，我们首先应当通过公式(4.9)和(4.10)计算水平初速度和竖直初速度。然后通过 4.11 计算出小球落地的时间。最后通过 4.7 计算出落地位移。具体的伪代码如下所示。注意在用 trigonometric 函数之前，我们必须把角度转化为弧度。

```
Create and initialize an array to hold ranges
for ii = 1:91
    theta ← ii – 1;
    vxo ← vo * cos(theta * conv);
    vyo ← vo * sin(theta * conv);
    max_time ← -2 * vyo / g;
    range(ii) ← vxo * max_time;
end
```

下一步，写出落地的表。伪代码如下

```
Write heading
for ii = 1:91
    theta ← ii – 1;
    print theta and range;
end
```

我们可以用 max 函数计算最大落地位移。调用这个函数返回最大值和它的位置。伪代码如下

```
[maxrange index] ← max(range);
Print out maximum range and angle (=index -1);
```

我们将用 for 循环嵌套来计算和画出抛物线。。为把所有抛物线都显示在屏幕上，在第一个抛物线的语句后，加上 hold on 命令。每画一个抛物线，就要用到一个 hold on 命令。在画最后一个抛物线时要用到 hold off 命令。我们将在抛物线上取 21 个重要的点，并找了这些的位置。我们将画出这些点。伪代码如下:

```
for ii = 5:10:85
% Get velocities and max time for this angle
theta ← ii – 1;
vxo ← vo * cos(theta * conv);
vyo ← vo * sin(theta * conv);
max_time ← -2 * vyo / g;
    Initialize x and y arrays
    for jj = 1:21
        time ←(jj – 1) * max_time / 20;
        x(time) ← vxo * time;
        y(time) ← vyo * time + 0.5 * g * time^2;
    end
    plot(x,y) with thin green lines
    Set "hold on" after first plot
end
Add titles and axis labels
```

最后，用不同的颜色且粗一点的线画出落地位移最大的抛物线。

```
vxo ← vo * cos(max_angle * conv);
vyo ← vo * sin(max_angle * conv);
max_time ← -2 * vyo / g;
Initialize x and y arrays
for jj = 1:21
    time ← (jj-1) * max_time / 20;
    x(jj) ← vxo * time;
    y(jj) ← vyo * time + 0.5 * g * time^2;
end
plot(x,y) with a thick red line
hold off
```

4. 转化 **MATLAB** 语句

```
% Script file: ball.m
%
% Purpose:
% This program calculates the distance traveled by a ball
% thrown at a specified angle "theta" and a specified
% velocity "vo" from a point on the surface of the Earth,
% ignoring air friction and the Earth's curvature. It
% calculates the angle yielding maximum range, and also
% plots selected trajectories.
%
```

```
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/10/97 S. J. Chapman Original code
%
% Define variables:
% conv Degrees to radians conv factor
% gravity Accel. due to gravity (m/s^2)
% ii, jj Loop index
% index Location of maximum range in array
% maxangle Angle that gives maximum range (deg)
% maxrange Maximum range (m)
% range Range for a particular angle (m)
% time Time(s)
% theta Initial angle (deg)
% traj_time Total trajectory time (s)
% vo Initial velocity (m/s)
% vxo Xcomponent of initial velocity (m/s)
% vyo Ycomponent of initial velocity (m/s)
% x Xposition of ball (m)
% y Yposition of ball (m)
% Constants
conv = pi / 180;       % Degreestoradians conversion factor
g = -9.81;             % Accel. due to gravity
vo = 20;               % Initial velocity
%Create an array to hold ranges
range = zeros(1,91); % Calculate maximum ranges
for ii = 1:91
    theta = ii - 1;
    vxo = vo * cos(theta*conv);
    vyo = vo * sin(theta*conv);
    traj_time = -2 * vyo / g;
    range(ii) = vxo * traj_time;
end
% Write out table of ranges
fprintf ('Range versus angle theta:\n');
for ii = 1:91
    theta = ii - 1;
    fprintf(' %2d %8.4f\n',theta, range(ii));
end
% Calculate the maximum range and angle
[maxrange index] = max(range);
maxangle = index - 1;
fprintf ('\nMax range is %8.4f at %2d degrees.\n',maxrange, maxangle);
% Now plot the trajectories
for ii = 5:10:85
    % Get velocities and max time for this angle
    theta = ii;
    vxo = vo * cos(theta*conv);
    vyo = vo * sin(theta*conv);
    traj_time = -2 * vyo / g;
    % Calculate the (x,y) positions
    x = zeros(1,21);
    y = zeros(1,21);
    for jj = 1:21
        time = (jj - 1) * traj_time/20;
        x(jj) = vxo * time;
```

```
            y(jj) = vyo * time + 0.5 * g * time^2;
        end
        plot(x,y,'b');
        if ii == 5
        hold on;
end
end
% Add titles and axis lables
title ('\bfTrajectory of Ball vs Initial Angle \theta');
xlabel ('\bf\itx \rm\bf(meters)');
ylabel ('\bf\ity \rm\bf(meters)');
axis ([0 45 0 25]);
grid on;
% Now plot the max range trajectory
vxo = vo * cos(maxangle*conv);
vyo = vo * sin(maxangle*conv);
traj_time = -2 * vyo / g;
% Calculate the (x,y) positions
x = zeros(1,21);
y = zeros(1,21);
for jj = 1:21
        time = (jj - 1) * traj_time/20;
        x(jj) = vxo * time;
        y(jj) = vyo * time + 0.5 * g * time^2;
end
plot(x,y,'r','LineWidth',3.0);
hold off
```

5. 检测程序

为了检测这个程序，我们计算手动计算了一些值，用来程序的输出结果作比较。

θ	$v_{xo}=v_o\cos\theta$	$v_{yo}=v_o\sin\theta$	$t_2=-2v_{yo}/g$	$x=v_{xo}t_2$
0°	20m/s	0	0	0
5°	19.92m/s	1.74m/s	0.355s	7.08m
40°	15.32m/s	12.86m/s	2.621s	40.15m
45°	14.14m/s	14.14m/s	2.883s	40.77m

当 ball 程序运行时，将 91 行含有角度和位移的结果。为了节省空间我们只打印其中的一部分。

```
>> ball
Range versus angle theta:
   0     0.0000
   1     1.4230
   2     2.8443
   3     4.2621
   4     5.6747
   5     7.0805
...
  40    40.1553
  41    40.3779
  42    40.5514
  43    40.6754
  44    40.7499
  45    40.7747
  46    40.7499
  47    40.6754
  48    40.5514
  49    40.3779
```

```
    50    40.1553
   ...
    85     7.0805
    86     5.6747
    87     4.2621
    88     2.8443
    89     1.4230
    90     0.0000

   Max range is    40.7747 at 45 degrees.
```

画图的结果为图 4.3。程序运算结果与手动运算结相符。当角度为 45 度时，位移最大。

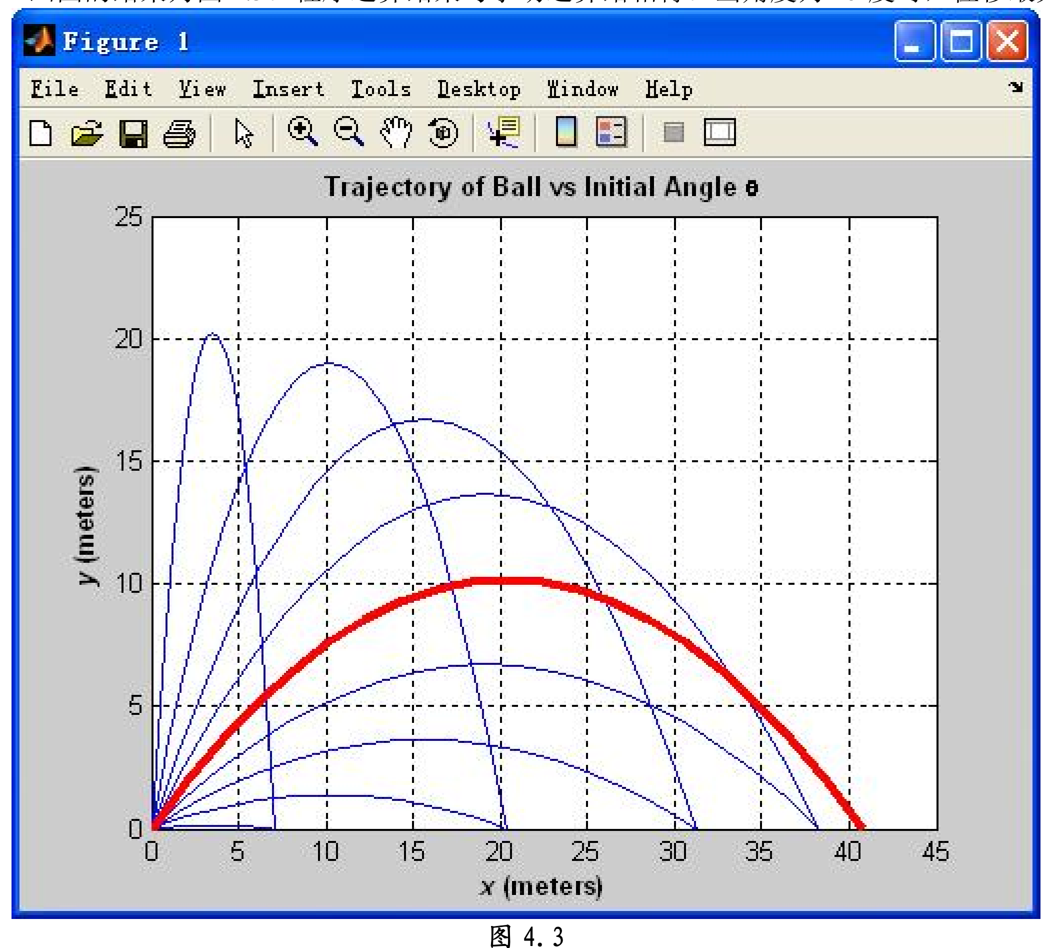


图 4.3

This example uses several of the plotting capabilities that we introduced in Chapter 3. It uses the axis command to set the range of data to display, the hold command to allow multiple plots to be placed on the same axes, the LineWidth property to set the width of the line corresponding to the maximum-range trajectory, and escape sequences to create the desired title and x- and y-axis labels.

这个例子用到在第三章中介绍的许多画图功能。我们用 axis 命令来显示水平位移，用 hold 命令让多幅图象在同一坐标系出现。用 linewidth 属性调整曲线的宽度。用转义序列创建所需的标题以及 x，y 坐标轴的标签。

但是这个程序不是最高效的，因为许多的循环可以用向量算法代替。练习题 4.11 将要求你重写并改进 ball.m。

4.5 总结

在 **MATLAB** 中有两种基本的循环形式，while 循环和 for 循环。while 循环中，代码
的重复的次数是不能确定的，只要满足用户定义的条件，重复就进行下去。相对地，在 for 循环中，代码的重复次数是确定的，在循环开始之前，我们就知道代码重复的次数了。在两种循环中均可使用 break 语句以跳出循环。

4.5.1 好的编程习惯总结

在有选择结构和循环结构的编程中，要遵循以下的编程指导思想.如果你长期坚持这些原则，你的代码将会有很少的错误，有了错误也易于修改，而且在以后修改程序时，也使别人易于理解.

1.对于 for 循环体总是要缩进两个或更多空格，以增强程序的可读性。

2.在循环体中绝不修改循环指数的值。

3.在循环执行开始之前，总是要预先分配一个数组。这样能大大增加循环运行的速度

4.如果用可能的话，可用逻辑函数选择数组中的元素。如果逻辑数组进行运算，要比循环快得多。

5.如果用可能的话，可用逻辑函数选择数组中的元素。如果逻辑数组进行运算，要比循环快得多。

4.5.2 MATLAB 总结

下面的总结列举了本章出现的所有特殊符号，命令和函数，后面跟的是简短的描述。

break	break 语句可以中止循环的执行和跳到 end 后面的第一句执行
continue	continue 语句只中止本次循环，然后返回循环的顶部。
for 循环	在 for 循环中，代码的重复次数是确定的
tic 函数	复位内建计时器
toc 函数	从最后一次调用 tic 以秒开始计时
while 循环	while 循环中，代码的重复的次数是不能确定的，只要满足用户定义的条件，重复就进行下去

4.6 练习

4.1 编写 **MATLAB** 语句计算 y(t)的值

$$y(t)=\begin{cases}-3t^2+5 & t\geq 0\\ 3t^2+5 & t<0\end{cases}$$

已知 t 从-9 到 9 每隔 0.5 取一次值。运用循环和选择语句进行计算。

4.2 用向量算法解决练习 4.1。

4.3 编写 **MATLAB** 语句计算并打印出 1 到 50 之间所有整数的平方。创建一个包含有每一个整数和他相应平方的和，注意在每一列加上合适的标签。

4.4 在 M 文件中编写程序，$y(x)=x^2-3x+2$，x 从 0.1 到 3 每隔 0.1 取一次值。用两种算法，一种是应用 for 循环，另一种是用数组。用 3point 粗的红虚线画出产生的函数。

4.5 在 M 文件中编写程序，计算阶乘 N!。注意 0!，如果 N<0 则报告出错。

4.6 检测下面的 for 语句，确定循环运行的次数。

a. for ii = -32768:32767
b. for ii = 32768:32767
c. for kk = 2:4:3
d. for jj = ones(5,5)

4.7 检测下面的 for 循环，确定每一次 for 循环结束的时侯 ires 的值。和每个 for 循环的次数。

```
a. ires = 0;
   for index = -10:10
       ires = ires + 1;
   end

b. ires = 0;
   for index = 10:-2:4
      if index == 0
         continue
      end
      ires = ires + index;
   end

c. ires = 0;
   for index = 10:-2:4
      if index == 0
         break;
      end
      ires = ires + index;
   end

d. ires = 0;
   for index1 = 10:-2:4
      for index2 = 2:2:index1
         if index2 ==6
            break;
         end
         ires = ires + index2;
      end
   end
```

4.8 检测下面的 while 循环，确定每一次 while 循环结束的时侯 ires 的值。和每个 while 循环的次数。

```
a. ires = i;
   while mod(ires,10) ~= 0
      ires = ires + 1;
   end

b. ires = 2;
   while ires <= 200
       ires = ires^2
   end

c. ires = 2;
   while ires > 200
      ires = ires^2;
end
```

4.9 当下面的语句执行后，数组 arr1 的结果是多少。

```
a. arr1 = [1 2 3 4; 5 6 7 8; 9 10 11 12];
   mask = mod(arr1,2) == 0;
   arr1(mask) = -arr1(mask);
```

```
b. arr1 = [1 2 3 4; 5 6 7 8; 9 10 11 12];
   arr2 = arr1 <= 5;
   arr1(arr2) = 0;
   arr1(~arr2) = arr1(~arr2) .^2;
```

4.10 如何将一个普通的数字数组带有逻辑性?如何去除数字数组的逻辑性?

4.11 修改例 4.8 中的 ball 程序，有向量算法代替内部 for 循环。

4.12 修改例 4.8 中的 ball 程序，在适当的位置读取重力加速度。并计算出最大的水平位移。修改成功后，分别用加速度-9.8m/s^2，-9.7m/s^2，-9.6m/s^2 进行运算。重力加速度的减小将会对小球的落地位移产生什么样的影响。重力加速度的减小会不会对掷球的最佳角度产生影响。

4.13 修改例 4.8 中的 ball 程序，读取小球的初速度。修改成功后，分别用加 10m/s，20m/s，30m/s 进行运算。改变初速度将会对小球的落地位移产生什么样的影响，对最佳抛射角度有没有影响?

4.14 例 4.7 中的程序 lsqfit 在输入数据之前，先要读取要输入数据的个数。我们要修改这个程序使之能够用 while 循环读取任意数量的数据，当用户敲击回车键时结束读取。用例 4。7 中的数据检测你的程序。(提示当用户按下回车键而不输入其他的任何值，input 函数将会返回一个空数组。你能用函数判断一个数组是不是空数组，当它为 true 时停止读取数据。)

4.15 修改 4.7 中的程序 lsqfit，使它能够从 input1.dat 文件中读取它的输入值。文件中的数据是以行组织的，每一行都有一对(x，y)，如下所示:

```
1.1   2.2
2.2   3.3
...
```

用例 4.7 中的数据检测你的程序。(提示:我们用 load 命令从 input1 数组读数据。然后把 input1 的第一列赋值于数组 x，把 input1 的第二列赋值于数组 y)。

4.16 **MATLAB** Least-Squares Fit Function. **MATLAB** includes a standard function that performs a least-squares fit to a polynormal. Function polyfit calculates the least-squares fit of a data set to a polynormal of order N

MATLAB 最小二乘匹配函数。**MATLAB** 包括一个标准函数，对多项式进行最小二乘匹配运算。函数 polyfit 用一系列数据对 *N* 阶多项式进行最小二乘匹配运算。

$$p(x)=a_nx^n + a_{n-1}x^{n-1} + ... + a_1x + a_0 \qquad (4.12)$$

其中 *N* 是任意大于等于 1。当 *N*=1 时，得到的是直线方程，带有斜率 a_1 和 *y* 轴截距 a_0，这个的形式如下：

p = polyfit(x,y,n)

x，y 代表向量 x，y，n 代表阶数。

应用函数 polyfit 修改例 4.7 中的 lsqfit 程序

4.17 在例 4.3 中，已知年月日，计算相应的 thedayofyear。在这个程序中，并没有检测是否输入了正确的年月日，它能接收无效的月和日，并产生无意义的结果。修改你的程序使之只能输入有效的年月日。如果输入的值无效，则提示用户出错，并且跳出执行。我们要求年应当大于 0，月只能是 1 到 12 之间的整数。日只能 1 到那一月的最大数之间的整数。用 switch 结构检查日是否正确。

4.18 编写 **MATLAB** 程序计算下列函数的值

$$y(x) = \ln\frac{1}{1-x}$$

x 为任意有意义的值，ln 为自然对数。用 while 循环编写程序，可以重复地输入合法的 x 的值并计算相应的函数值。当一个非法的 x 输入，则中止程序。（所有 x≥1 的数均为非法值）

4.19 斐波那契数列。含有 n 个数的斐波那契数列的定义如下：

f(1) = 1

f(2) = 2

f(n) = f(n-1) + f(n-2)

所以 $f(3)=f(2)+f(1)=2+1=3$，还有更多的数。在 M 文件中编写一程序，计算并写斐波那契数列中第 n（$n>2$）个数的值，n 由用户输入。用 for 循环进行计算。

4.20 通过二极管的电流

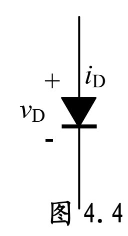


图 4.4

如图 4.4 所示的半导体二极管，通过它的电流可由下面的公式得到

$$i_D = I_o(e^{\frac{qv_D}{kT}} - 1) \tag{4.13}$$

v_D 代表管压降，单位为伏（V）

i_D 代表通过二极管的电流，单位为安（A）

i_o 代表反向饱和电流

q 代表电子的电量 1.602×10^{-19} 库仑（C）

k 代表玻尔兹曼常数 1.38×10^{-23}J/K

T 代表开尔文温度

4.21

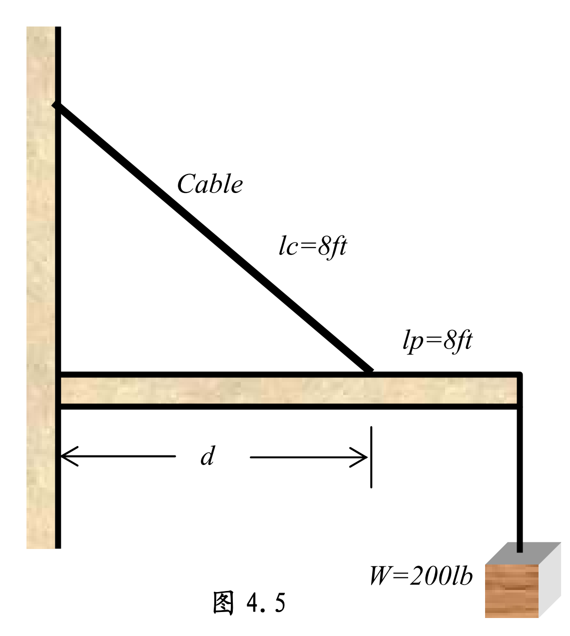


图 4.5

轻绳上的拉力。一重 200 英磅的物体被固定在一水平杆的末端。如图 4.5 所示这一水平杆由一轻绳固定。绳子上的拉力为

$$T = \frac{W \cdot lc \cdot lp}{d\sqrt{lp^2 - d^2}} \tag{4.14}$$

T 代表绳子的拉力，W 代表物体的重量，lp 代表杆的长度，lc 为绳长，d 代表绳与杆的结点到墙面的距离。编写一个程序，以确定 d 为多大时，绳的拉力最小。为达此目的，d 应从 1 英尺到 7 英尺，每隔 1 英尺取一次值，并找出使拉力最小的 d。

4.22 病毒的繁殖。假设某生物学家做实验，测试一种特殊病毒在不同的培养基下的繁殖速率。实验表明 A 培养基中的病毒每 60 分钟复制一次，A 培养基中的病毒每 90 分钟复制一次。假设在两个培养基中开始的时候各有一个病毒。编写一个程序，计算 24 小时之内每隔三小时病毒在各自培养基中的个数，并画出图象。画两个图，一个用线性坐标，一个用线性对数坐标。24 小时之后两培养基中的病毒的数目分别是多少？

4.23 分贝

工程师们经常用分贝或 dB 来描述两功率之比。1dB 的定义如下，

$$dB = 10\log_{10}\frac{P_2}{P_1} \qquad (2.13)$$

P_2 是已测量的功率，P_1 代表参考功率。假设参考功率 P_1 是 1 瓦。P_2 从 1 到 20 瓦每隔 0.5 瓦取一次值，编写程序，计算相应的 *dB* 值，并画出 *dB*-P_2

4.24 图象

几何平均数(geometric mean)。数列中的元素从 x_1 到 x_n，它们的几何平均数的定义如下

$$\text{geometric mean} = \sqrt[n]{x_1 x_2 x_3 \ldots x_n} \qquad (4.16)$$

编写一个程序，它能接受任意个数的正输入值，并计算它们的算术平均数和几何平均数。用 while 循环读取输入值，当输入一个负数中止输入数据。计算数列 10，5，2，5 的算术平均数和几何平均数，用以检测程序。

4.25 均方根平均数（rmsaverage）。均方根平均数是另一种计算数据平均数的方法。它的定义如下

$$\text{rms average} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \qquad (4.17)$$

编写一个程序，它能接受任意个数的正输入值，并计算它们的算太平均数和几何平均数。用 while 循环读取输入值，当输入一个负数中止输入数据。计算数列 10，5，2，5 的均方根平均数，用以检测程序。

4.26 调和均数(harmonic mean)。调和均数也是另一种计算数据平均数的方法。它的定义如下

$$\text{harmonic mean} = \frac{N}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_N}} \qquad (4.18)$$

编写一个程序，它能接受任意个数的正输入值，并计算它们的算太平均数和几何平均数。用 while 循环读取输入值，当输入一个负数中止输入数据。计算数列 10，5，2，5 的调和均数，用以检测程序。

4.27 编写一个程序，能够计算一系列正数的算术平均数，几何平均数，均方根平均数，调和均数。可使用任意算法读取输入值。用下列数测试你的程序。

```
a. 4, 4, 4, 4, 4, 4, 4
b. 4, 3, 4, 5, 4, 3, 5
c. 4, 1, 4, 7, 4, 1, 7
d. 1, 2, 3, 4, 5, 6, 7
```

4.28 平均无故障时间（Mean Time Between Failure）。电器设备的可靠性是用平均无故障时间来度量的，它是指设备正常工作的平均时间。对包含许多电器设备的大系统来说，它的平均无障碍时间取决于每一个部分平均无障碍时间，我们可以通过第一个部分的平均无障碍时间计算整体的平均无障碍时间。如果系统的内部结构如图 4.6 所示，那么全局的 MTBF 为

$$\text{MTBF}_{sys} = \frac{1}{\frac{1}{\text{MTBF}_1} + \frac{1}{\text{MTBF}_2} + \ldots + \frac{1}{\text{MTBF}_n}} \qquad (4.19)$$

编写一个程序，读取每一个部分的平均无障碍时间，然后计算出全局的 MTBF。用下面的数据测试你的程序，subsystem1，2，3，4 的平均无障碍时间分别为 2000 小时，800 小时，3000 小时，5000 小时。

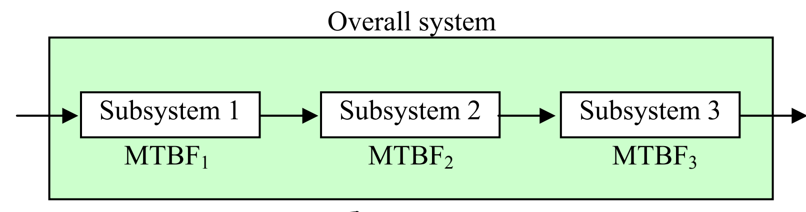

Overall system
Subsystem 1
Subsystem 2
Subsystem 3
MTBF1
MTBF2
MTBF3

图 4.6 MTBF

第五章 自定义函数

在第三章中，我们强调了好的编程习惯的重要性。我们进行开发的基本手段是自上而下的编程方法。在自上而下的编程方法中，它开始于对所要解决问题的精确陈述和定义输入量和输出量。下一步，我们在大面上进行算法的描述，然后把算法分解成一个一个的子问题。再然后，程序员把这一个个子问题进行再一次的分解，直到分解成简单而且能够清晰理解的伪代码。最后把伪代码转化为 **MATLAB** 代码。

尽管我们在前面的例子中，按照上面的步骤进行了编程。但是产生的结果在某种程度上还是受限制的。因为我们必须把每一个子问题产生的 **MATLAB** 代码嵌入到一个单独的大程序中。在嵌入之前我们无法对每一次子问题的代码进行独立地验证和测试。

幸运的是，**MATLAB** 有一个专门的机制，在建立最终的程序之前用于独立地开发与调试每一个子程序。每一个子程序都可以独立函数的形式进行编程，在这个程序中，每一个函数都能独立地检测与调试，而不受其他子程序的影响。良好的函数可以大大提高编程的效率。它的好处如下：

1.子程序的独立检测

每一个子程序都可以当作一个独立的单元来编写。在把子程序联合成一个的大程序之前，我们必须检测每一个子程序以保证它运转的正确性。这一步就是我们熟知的单元检测。在最后的程序建立之前，它排除了大量的问题。

2.代码的可复用性

在许多的情况下，一个基本的子程序可应用在程序的许多地方。例如，在一个程序的许多地方，要求对一系列按由低到高的顺序进行排序。你可以编一个函数进行排序，然后当再次需要排序时可以调用这个函数。可重用性代码有两大好处：它大大提高了整体编程效率，它更易于调试，因为上面的排序函数只需要调试一次。

3.远离意外副作用

函数通过输入参数列表（input argument list）从程序中读取输入值，通过输出参数列表（output argument list）给程序返回结果。程序中，只有在输入参数列表中的变量才能被函数利用。函数中，只有输出参数列表中的变量才能被程序利用。这是非常重要的，因为在一个函数中的突发性编程错误只会发生错误的函数的变量。一旦一个大程序编写并发行，它还要面临的问题就是维护。程序的维护包括修补错误，修改程序以适应新或未知的环境。作维护工作的程序员在一般情况下不会是程序的原作者。如果程序编写的不好，改动一处代码就可能对程序全局产生负面影响。这种情况的发生，可能是因为变量在其他部分被重新定义或利用。如果程序员改变这个变量，可能会导致后面的程序无法使用。

好的函数的应用可以通过数据隐藏使问题最小化。在主函数中的变量在函数中是不可见的（除了在输入变量列表中的变量），在主程序中的变量不能被函数任意修改。所以在函数中改变变量或发生错误不会在程序的其他部分发生意外的副作用。

好的编程习惯

把大的程序分解成函数，有很多的好处，例如，程序部分的独立检测，代码的可复用性，避免意想不到的错误。

5.1 MATLAB 函数简介

到目前为止，我们看到的所有的 M 文件都是脚本文件。脚本文件只是用于存储 **MATLAB** 语句。当一个脚本文件被执行时，和直接在命令窗口中直接键入 **MATLAB** 语句所产生的结果是一样的。脚本文件分享命令窗口中的工作区，所以所有的在脚本文件运行之前定义的变量都可以在脚本文件中运行，所有在脚本文件中创建的变量在脚本文件运行之后仍然存在工作区。一个脚本文件没有输入参数，也不返回结果。但是所有脚本文件可以通过存于工作区中的数据进行交互。

相对地，**MATLAB** 函数是一种特殊形式的 M 文件，它运行在独立的工作区。它通过输入参数列表接受输入数据，它通过输出参数列表返回结果给输出参数列表。**MATLAB** 函数的基本形式如下：

```
function [outarg1, outarg2, ...] = fname(inarg1, inarg2, ...)
%H1 comment line
%Other comment lines
...
(Executable code)
...
(return)
```

function 语句标志着这个函数的开始。它指定了函数的名称和输入输出列表。输入函数列表显示在函数名后面的括号中。输出函数列表显示在等号左边的中括号中。（如果只有一个输出参数，中括号可以省略。）

输入参数列表是名字的列表，这些名字代表从调用者到函数的值。这些名字被称作形参。当函数被调用时，它们只是从调用者得来实际变量的占位符而已。相似地，输出参数列表也形参组成，当函数结束运行时，这些形参是返回到调用者的值的占位符。

在一个表达式中，调用一个函数需要用到实参列表。在命令窗口直接（或在脚本文件中，另一个函数中）键入函数的名字就可以调用这个函数了。当调用一个函数时，第一个实参的值用在第一个形参的位置，而且其余的形参和实参都一一对应。

函数的执行从函数的顶部开始，结束于 return 语句或函数的终点。因为在函数执行到结尾就会结束，所以 return 语句在大部分的程序中没有必要使用。在输出参数列表中每一个项目都必须出现在 function 语句中的左边。当函数返回时，存储于输出函数列表的值就会返回给调用者，用于下一步的运算。

在一个函数中的初始注释行有特定的目的。在 function 语句的第一个行注释被称为 **H1 注释行**。它应当是对本函数功能的总结。这一行的重要性在于，通过 lookfor 命令它能被搜索到并显示出来。从 **H1 注释行**到第一个空行或第一个可执行性语句可以通过 help 命令或帮助窗口搜索到。它们则应包含如何使用这个函数的简单总结。

下面是一个自定义函数的简单例子。函数 dist2 用于计算笛卡尔坐标系中点（x_1，y_1）与点（x_2，y_2）之间的距离。(把以下代码保存成 dist2.m 文件)

```
function distance = dist2 (x1, y1, x2, y2)
%DIST2 Calculate the distance between two point
% Function DIST2 calculates the distance between
% two points (x1, y1) and (x2,y2) in a cartesian
% coordinate system.
%
% Calling sequence:
%      res = dist2(x1, y1, x2, y2)
```

```
%
% Define variables:
% x1          --x-position of point 1
% y1          --y-position of point 1
% x2          --x-position of point 2
% y2          --y-position of point 2
% distance    --Distance between points
%
% Record of revisions:
%       Date           Pragrammer        Description of change
%       ========       ==========        ================
%       12/15/98  S.J.Chapman        Original code
%
% Calculate distance.
distance = sqrt((x2-x1).^2 + (y2-y1).^2);
```

这个函数有 4 个输入参数各和 1 个输出参数。一个简单的利用这个函数的例子显示如下：

```
%   Script file: test_dist2.m
%
%   Purpose:
%       This program test2 function dist2.
%
%   Record of revisions:
%       Date           Pragrammer        Description of change
%       ========       ==========        ================
%       12/15/98  S.J.Chapman        Original code
%
%   Define variables:
%   ax    --x-position of point a
%   ay    --y-position of point a
%   bx    --x-position of point b
%   by    --x-position of point b
%
%   Get input data.
disp('Calculate the distance between two points:');
ax = input ('Enter x value of point a:');
ay = input ('Enter y value of point a:');
bx = input ('Enter x value of point b:');
by = input ('Enter y value of point b:');

%   Evaluate function
result = dist2 (ax, ay, bx, by);
%   Write out result.
fprintf('The distance between points a and b is %f \n', result);
```

当脚本文件被执行时，它的结果显示如下：

```
>> test_dist2
Calculate the distance between two points:
Enter x value of point a:1
Enter y value of point a:1
Enter x value of point b:4
Enter y value of point b:5
The distance between points a and b is 5.000000
```

通过手动运算我们可知程序运算的结果是正确的。

函数 dist2 也支持 **MATLAB** 帮助子系统。如果你键入“help dist2”，将会得到的结果是：

```
>> help dist2
```

```
 DIST2 Calculate the distance between two point
  Function DIST2 calculates the distance between
  two points (x1, y1) and (x2,y2) in a cartesian
  coordinate system.

  Calling sequence:
     res = dist2(x1, y1, x2, y2)

  Define variables:
  x1          --x-position of point 1
  y1          --y-position of point 1
  x2          --x-position of point 2
  y2          --y-position of point 2
  distance    --Distance between points

  Record of revisions:
        Date          Pragrammer        Description of change
        ========      ===========       =================
        12/15/98  S.J.Chapman        Original code

  Calculate distance.
```

相似地，键入“lookfor dist2”后将会产生如下的结果：

```
>> lookfor dist2
DIST2 Calculate the distance between two point
test_dist2.m: %      Script file: test_dist2.m
>> lookfor distance
DIST2 Calculate the distance between two point
```

为了仔细观察工作区在函数执行前后的变化，我们将在 **MATLAB** 调试器中加载函数 dist2 和脚本文件 test_dist2。在函数加载前，加载中，加载后设置断点（如图 5.1 所示）。

当程序中止在函数调用之前的断点，它的工作区如图 5.2(a)所示。注意工作区中只有变量 ax，ay，bx 和 by。当程序中止在函数调用过程中的断点，它的工作区如图 5.2(b)所示。注意工作区中只有变量 x1，x2，y1，y2 和 distance。当程序中止在函数调用后的断点，它的工作区如图 5.2(c)所示。注意工作区中原来的变量又重复出现，再加上函数返回的变量 result。这些图显示了 **MATLAB** 调用 M 文件的过程中工作区的变化。

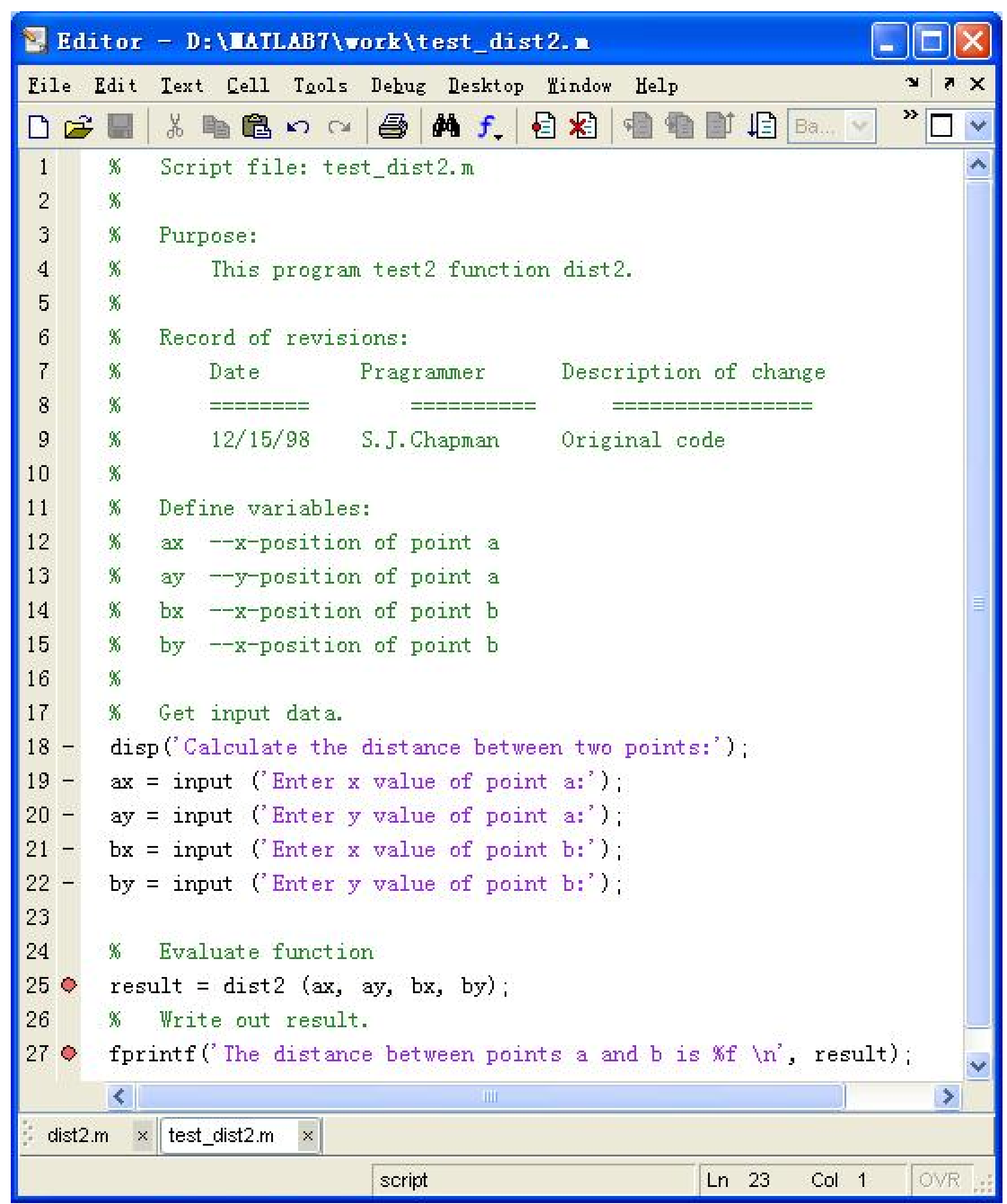


图 5.1 M 文件和函数 dist2 将会被加载到调试器，在函数调用前，调用过程中，调用后设置合适断点

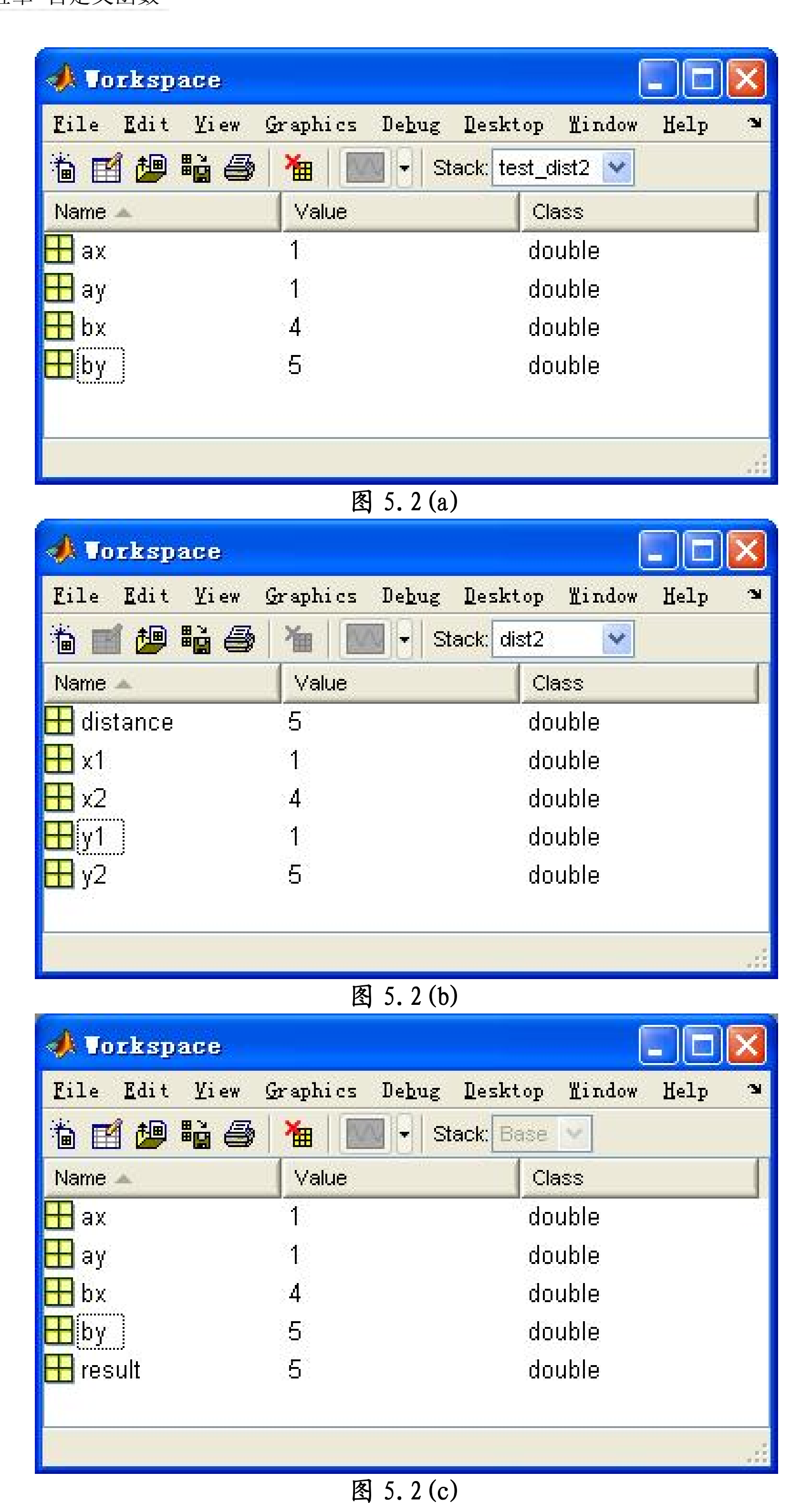


图 5.2(a)

图 5.2(b)

图 5.2(c)

图 5.2(a)在函数调用之前的工作区（b）函数调用过程中的工作区（c）函数调用之后的工作区

5.2 在 MATLAB 中传递变量：按值传递机制

maltab 程序与它们函数之间的交互用是按值传递机制。当一个函数调用发生时，

MATLAB 将会复制实参生成一个副本，然后把它们传递给函数。这次复制是非常重要的，因为它意味着虽然函数修改了输入参数，但它并没有影响到调用者的原值。这个特性防止了因函数修改变量而导致的意想不到的严重错误。

这一特性将在下面的函数中得到说明。这个函数中有两个输入参数：a 和 b。在它的计算中，它修改了变量的值：

```
function out = sample(a, b)
fprintf('In Sample: a = %f, b = %f %f\n',a,b);
a = b(1) + 2*a;
b = a .* b;
out = a + b(1);
fprintf('In Sample: a = %f, b = %f %f \n',a,b);
```

下面是调用这个函数的检测程序：

```
a = 2; b = [6 4];
fprintf('Before sample: a = %f, b = %f %f\n', a, b);
out = sample(a, b);
fprintf('After sample: a = %f, b = %f %f\n',a,b);
fprintf('After sample: out = %f \n', out);
```

当这个程序被执行将产生如下的结果：

```
>> test_sample
Before sample: a = 2.000000, b = 6.000000 4.000000
In Sample: a = 2.000000, b = 6.000000 4.000000
In Sample: a = 10.000000, b = 60.000000 40.000000
After sample: a = 2.000000, b = 6.000000 4.000000
After sample: out = 70.000000
```

注意，a 和 b 在函数 sample 内都改变了，但这些改变对调用函数中的值并没有任何的影响。

C 语言的使用者对按值传递机制比较熟悉，因为 C 应用它把标量值传递给函数。尽管 C 语言不能用按值传递机制传递数组，所以对在 C 语言函数中的形参数组进行意想不到的修改将会导致在调用程序时产生错误。**MATLAB** 改进了按值传递机制，既适于标量，又适应于数组（在 **MATLAB** 中参数传递过程中的执行要远比上面讨论中指出的要复杂的多。正如上面指出的，与按值传递相联系的复制将花去大量的时间，但是保护程序以至于不产生错误。实际上，**MATLAB** 用了两种方法，它先对函数的每一个参数进行分析，确定函数的那些参数进行了修改。如果函数没有修改这个参数，它将不会对此参数进行复制，而是简单地指向程序外面的外面的变量，如果函数修改了这个参数，那么这个复制就会被执行）。

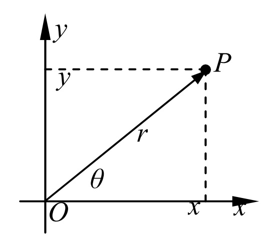


图 5.3 在笛卡尔平面内的一点 P，既可以用直角坐标系来描述，又可以有极坐标来描述

例 5.3

直角坐标与极坐标的转换

在笛卡尔平面上的一点的坐标既可以通过直角坐标（x，y）来描述，也可以通过极坐标（r，θ）来描述，如图 5.3 所示。两套坐标体系的关系如下式所示：

$$x = r\cos\theta \qquad (5.1)$$

$$y = r\sin\theta \qquad (5.2)$$

$$r = \sqrt{x^2 + y^2} \qquad (5.3)$$

$$\theta = \tan^{-1}\frac{y}{x} \qquad (5.4)$$

编写两个函数 rect2polar 和 polar2rect，用来实现两坐标体系的转换。其中 θ 单位于为度。

答案：

我们现在应用标准的问题解决方法来创建函数。注意 matab 的 trigonometric(三角)函数的参数的单位是弧度。所以在解决这个问题时，我们必须把度转化为弧度，反之亦然。基本的度与弧度的转换关系如下：

$$180^{o} = \pi \text{ radians} \qquad (5.5)$$

1.陈述问题对这个问题的简单陈述为：

编写一个函数，把直角坐标系描的笛卡尔平面内的一个坐标转化对应的极坐标，角 θ 的单位为度。反之，把极坐标系内的一个坐标转化对直角坐标。角 θ 的单位也为度。

2.定义输入输出量

函数 rect2polar 的输入量是直角坐标系（x，y）中的一个点。这个函数的输出量是极坐标（r，θ）中的一个点。函数 polar2rect 的输入量是极坐标（r，θ）中的一个点。此函数的输出量是直角坐标系（x，y）中的一个点。

3.定义算法

这些函数是非常简单的，所以我们能直接的写出它们的伪代码。函数 polar2rect 的伪代码如下：

```
x ← r * cos(theta * pi/180)
y ← r * sin(theta * pi/180)
```

函数 rect2polar 的伪代码将会用到函数 atan2，因为函数的取值范围覆盖了笛卡尔平面的所有象限。（通过 **MATLAB** 的帮助系统查找这个函数。）

4.把算法转化为 **MATLAB** 语句

函数 polar2rect 的 **MATLAB** 代码如下所示

```
function [x, y] = polar2rect(r, theta)
%POLAR2RECT Convert rectangular to polar coordinates
%    Function POLAR2RECT accepts the polar coordinates
%    (r, theta), where theta is expressed in degrees,
% and converts them into the rectangular coordinates (x, y)
%
%    Calling sequence:
%          [x, y] = polar2rect(r, theta)

%    Define variables:
%    r              --Length of polar vector
%    theta          --Angle of vector in degrees
%    x              --x-position of point
%    y              --y-position of point

%    Record of revisions:
%          Date       Programmer         Description of change
%          ======       ===========       ================
%          09/19/00  S.J.Chapman         Original code
x = r * cos(theta * pi/180);
y = r * sin(theta * pi/180);
```

函数 rect2polar 的 **MATLAB** 代码如下所示

```
function [r, theta] = rect2polar(x, y)
%RECT2POLAR Convert rectangular to polar coordinates
%    Function RECT2POLAR accept the rectangular coordinates
%    (x, y) and converts them into the polar coordinates
%    (r, theta), where theta is expressed in degrees.
%
%    Calling sequence:
```

```
%       [r, theta] = rect2polar(x, y)
%
%   Define variables:
%   r           --Length of polar vector
%   theta       --Angle of vector in degrees
%   x           --x-position of point
%   y           --y-position of point
%
%   Record of revisions:
%       Date        Programmer      Descriptoin of change
%       ====        ==========      ================
%       09/19/00  S.J.Chapman       Original code

r = sqrt( x .^2 + y .^2);
theta = 180/pi * atan2(y, x);
```

注意这两个函数中都包含了帮助信息，所以在应用 **MATLAB** 的帮助子系统中，或使用 lookfor 命令中它们将正常的运作。

5.检测程序

为了检测这些程序，我们将在 **MATLAB** 命令窗口中直接运行它们。我们将用边长分别为 3，4，5 的三角进行检测，这个三角在初中我们就非常的熟悉了。在这个三角形中最小的角约为 36。87 度。我们将在四个象限对函数进行检测，以保证在任何情况下转换都是正确的。

```
>> [r, theta]=rect2polar(4,3)
r =
     5
theta =
   36.8699
>> [r, theta]=rect2polar(-4,3)
r =
     5
theta =
  143.1301
>> [r, theta]=rect2polar(-4,-3)
r =
     5
theta =
 -143.1301
>> [r, theta]=rect2polar(4,-3)
r =
     5
theta =
  -36.8699

>> [x, y]= polar2rect(5,36.8699)
x =
    4.0000
y =
    3.0000
>> [x, y]= polar2rect(5,-143.1301)
x =
   -4.0000
y =
   -3.0000
>> [x, y]= polar2rect(5,143.1301)
x =
```

```
    -4.0000
y =
     3.0000
>> [x, y]= polar2rect(5,-36.8699)
x =
     4.0000
y =
    -3.0000
```

在笛卡尔坐标的四个象限内得到的结果均是正确的。

例 5.2 数据排序

在许多的科研和工程应用中，随机输入一组数据并对它进行由低到高排序或由高到低进行排序是十分必要的。假设你是一个动物学家，你正在研究大量的动物，并想要鉴定这些动物最大的 5%。

解决这个问题的最直接的方法是对这些动物的大小按照降序进行排列，取其前 5%。对数据进行升序或降序排列似乎是一件非常容易的工作。毕竟，我们经常作这样的事。有一个简单的例子，把数据（10，3，6，4，9）按升序排列成（3，4，6，9，10）。我们应当怎样做呢。我们首先应当浏览整个输入数据列表（10，3，6，4，9）找出其中的最小值（3），然后浏览剩下的输入数据（10，6，4，9）并找到下一个最小值（4），然后继续重复上面的步骤，直到所有的列表中的所有数都能排完。

实际上，排序是一个非常困难的工作。当值的个数增加时，用上面简单的排序方法进行运算所消耗的时间将会迅速增加，因为每排一个数就要浏览一个遍输入值。对于大的数据集合，这个方法因太耗时，而不用。更糟糕的是，如果有大量的数据占有计算机的大部分内存我们将如何排序。开发大数据集合的高效排序技术是一个相当活跃的研究领域，它已经成为了一个新的科目。

在这个例子中，我们将尽可能简单的算法来说明排序的内容。这个最简单的算法叫做选择性排序（selection sort）。它只是对应上面描述方法的计算机执行。选择性排序的基本算法如下：

1.浏览被排序数的列表，并找出其中的最小值。把最小值与排在最前面的数进行交换。如要排在最前面的数就是这个数表最小值，什么也不用做。

2.从这个数据列表的第二个数开始浏览找到第二个最小的数。把这个数与当前排在第二个数进行交换。如果当前排在第二位的数就是下一个最小值，那么什么也不用做。

3.从数据列表的第三个数开始找到第三个最小的数。把这个数与当前排在第三个数进行交换。如果当前排在第三位的数就是第三个最小值，那么什么也不用做。

4.重复以上步骤直至最后一位置排完。当最后一个位置排完后，排序结束。

注意:如果我们要对 N 个数进行排序，这个排序算法要进行 N-1 次浏览才能完成排序。

这个步骤的说明如图 5.4 所示。因为有 5 个数进行排序，所以要对数据进行 4 次浏览。首先对整个数据列表进行浏览，得到最小值 3，把 3 置于第一位，故与 10 进行交换。从第二位开始浏览，得到第二个最小值 4，与 10 交换。从第三位进行浏览，得到最小值 6，6 恰在第三位上，不用交换。从第四位开始浏览，得到最小值 9，与排在第 4 位的 10 交换。排序结束。

性能提示

选择性编程算法是一种极易理解的编程算法，但它的效率是极低的。我们绝不能用它进行大数据集合的排序（例如含有 1000 个元素的数组）。这个几年里，计算机专家已经发展了许多高效的排序算法。内置于 MATLAB 的 sort 和 sortrows 函数是非常高效的，在实际工作中我们应当应用这些函数。

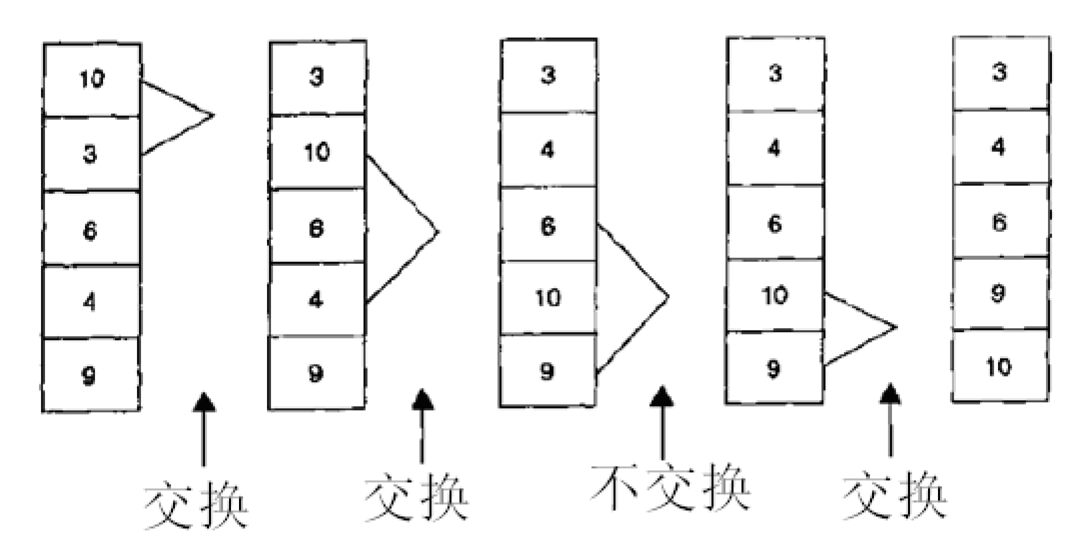


图 5.4 选择性排序的一个简单例子。

我们将开发一个程序，读取从命令窗口读取一个数据集，对它进行升序排列，并出排序后的结果。这个排序将会由独立的自定义函数来完成。

答案：

这个程序必须提示使用者提供输入数据，对其进行排序，并输出排序结要。这个程序的设计过程如下：

1.陈述问题我们刚才没有指定要排序的数据类型。如果数据是数字，那么问题的陈述如下。开发一个程序，它能够读取在命令窗口中输入的任意类型的数字，用独立的自定义函数对读取的值进行排序，并在命令窗口写出排序结果。

2.定义输入输出量这个程序的输入值是在命令窗口键入的数字值。这个程序的输出量是写在命令窗口中的排序结果。

3.设计算法这个问题可以分解为三大步：

```
Read the input data into an array
Sort the data in ascending order
Write the sorted data
```

第一大步是读取数据。我们必须提示使用者输入输入数据的个数，然后读取数据。因为我们知道所要读取的数的确切个数，所以可以用 for 循环主读取合适的数据。它的伪代码如下：

```
Prompt user for the number of data values
Read the number of data values
Preallocate an input array
for ii = 1:number of values
    Prompt for next value
    Read value
end
```

下一步，我们必须要用独立的函数对数据进行排序。我们需要对数据进行 naval-1 次浏览，每一次找出一个最小值。我们将用一个指针来寻找每一次浏览的最小值。一量最小值被找到，如果它不在列表的顶端，它就与列表顶端的元素进行交换。伪代码如下：

```
for ii = 1:nvals -1
% Find the minimum value in a(ii) through a(nvals)
    iptr ← ii
    for jj = ii +1 to nvals
        if a(jj) < a(iptr)
            iptr ← a(iptr)
        end
    end
```

```
% iptr now points to the min value, so swap a(iptr)
% with a(ii) if iptr ~= ii.
    if ii ~= iptr
        temp ← a(ii)
        a(ii) ← a(iptr)
        a(iptr) ← temp
    end
end
```

最后一步是输出排序结果。这个步骤的伪代码不需要重复。最终的伪代码是这三大步伪代码的联合。

4.把伪代码转化为 **MATLAB** 语言

选择性排序的 **MATLAB** 代码如下所示：

```
function out = ssort(a)
%SSORT Selection sort data in ascending order
% Function SSORT sorts a numeric data set into
% ascending order. Note that the selection sort
% is relatively inefficient. DO NOT USE THIS
% FUNCTION FOR LARGE DATA SETS. Use MATLAB's
% "sort" function instead.
% Define variables:
% a         --Input array to sort
% ii        --Index variable
% iptr      --Pointer to min value
% jj        --Index variable
% nvals     --Number of values in "a"
% out       --Sorted output array
% temp      --Temp variable for swapping
% Record of revisions:
% Date        Programmer        Description of change
% ====        ==========        =====================
% 12/19/98    S. J. Chapman     Original code
% Get the length of the array to sort
nvals = size(a,2);
% Sort the input array
for ii = 1:nvals-1
% Find the minimum value in a(ii) through a(n)
    iptr = ii;
    for jj = ii+1:nvals
        if a(jj) < a(iptr)
            iptr = jj;
        end
    end
% iptr now points to the minimum value, so swap a(iptr)
% with a(ii) if ii ~= iptr.
    if ii ~= iptr
        temp = a(ii);
        a(ii) = a(iptr);
        a(iptr) = temp;
    end
end
% Pass data back to caller
out = a;
```

调用选择性排序函数的程序如下：

```
% Script file: test_ssort.m
%
```

```
% Purpose:
% To read in an input data set, sort it into ascending
% order using the selection sort algorithm, and to
% write the sorted data to the Command window. This
% program calls function "ssort" to do the actual
% sorting.
%
% Record of revisions:
% Date          Programmer          Description of change
% ====          ==========          =====================
% 12/19/98   S. J. Chapman      Original code
%
% Define variables:
% array              --Input data array
% ii                 --Index variable
% nvals              --Numberof input values
% sorted             --Sorted data array
% Prompt for the number of values in the data set
nvals = input('Enter number of values to sort: ');
% Preallocate array
array = zeros(1,nvals);
% Get input values
for ii = 1:nvals
      %Prompt for next value
      string = ['Enter value ' int2str(ii) ': '];
      array(ii)=input(string);
end
% Now sort the data
sorted = ssort(array);
% Display the sorted result.
fprintf('\nSorted data:\n');
for ii = 1:nvals
      fprintf(' %8.4f \n',sorted(ii));
end
```

5.检测程序

为了检测这个程序，我们应当创建一个输入数据集，并运行这个程序。这个数据集应当包含由正负数混合组成，还有至少包括一个完全一样的值，以观察在这些情况下程序是否正常工作。

```
>> test_ssort
Enter number of values to sort: 6
Enter value 1: -5
Enter value 2: 4
Enter value 3: -2
Enter value 4: 3
Enter value 5: -2
Enter value 6: 0

Sorted data:
   -5.0000
   -2.0000
   -2.0000
    0.0000
    3.0000
    4.0000
```

对于我们检测的数据，程序给出了正确的结果。注意从正数到负数，还有重复值，这个程序工作正常。

5.3 选择性参数

许多的 **MATLAB** 函数都支持选择性输入参数和输出参数。例如，我们调用 plot 函数，输入参数既可以少到 2 个，也可以多到 7 个参数。从另一方面说，函数 max 既支持一个输出参数，也支持两个输出参数。如果只有一个输出参数，max 将会返回函数的最大值。如果有两个输出参数将会返回数组的最大值和最大值所在的位置。如何知道一个 **MATLAB** 函数有几个输入输出参数呢，以及函数相应的功能呢？

在 **MATLAB** 中有八种专门的函数用于获取关于选择性参数的信息和用于报告这些参数的错误。其中的六个函数我们在这里介绍，其余的两个我们将会在第七章讲单元数据类型时介绍。

nargin	这个函数返回调用这个函数时所需要的实际输入参数的个数
nargout	这个函数返回调用这个函数时所需要的实际输出参数的个数
nargchk	如要一个函数调用被调用时参数过多或过少，那么 nargchk 函数将返回一个标准错误信息
error	显示错误信息，并中止函数以免它产生这个错误。如果参数错误是致命的，这个函数将会被调用。
warning	显示警告信息并继续执行函数，如果参数错误不是致命的，执行还能继续，则这个将会被调用。
inputname	这个函数将会返回对于特定参数个数的实际变量名。

函数 nargin 和 nargout 只用在用户自定义函数中。当他们被调用时，这些函数将会分别返回实际输入、输出参数的个数。如果一个函数在被调用时含有过多或过少的参数，函数 nargchk 将会产生一个包含标准错误的字符串。此函数的语法如下：

```
message = nargchk(min_args, max_args, num_args);
```

其中 min_args 是指参数的最小个数，max_args 是指数的最大个数，num_args 是指参数的实际个数。如果参数的个数不在允许的范围，将会产生一个标准的错误信息。如果参数的个数在允许的范围之内，那么这个函数将返回一个空字符。

函数 error 是用于显示标准的错误信息和用于中止导致错误信息的自定义函数的一种标准方式。这个函数的语法是 error('msg')，其中 msg 是一个包含错误信息的字符串。当 error 函数执行，它将会中止当前函数，并返回到键盘输入状态，在命令窗中显示出错误信息。如果这个信息字符中为空，error 函数将什么也不做，当前函数继续执行。如果当前函数与线程数 nargchk 工作良好，当有错误发生时，error 将产生一个信息字符串，当没有错误时，error 将产生一个空字符。

函数 warning 是用于显示函数或线程数中的警告信息的一种标准方法。此函数的语法为 warning('msg')，其中 msg 是指含有警告信息的字符串。当执行 waring 函数时，它将在命令窗口显示警告信息，和列出警告出现的函数和线程数。如果信息子符串为空，warning 将什么也不做。在其他情况下，函数将继续执行。

当一个函数被调用时，inputname 函数将会返回实参的名字。inputname 函数的语法为

```
name = inputname(argno);
```

其中 argno 是参安息的个数。如果这个参数是一个变量，那么返回将只是变量名。如果参数是一个表达式，那么这个函数将会返回空字符。例如考虑下面的函数

```
function myfun(x, y, z)
name = inputname(2);
disp(['The second argument is named ' name]);
```

当这个函数被调用时，结果如下

```
>> myfun(dog,cat)
The second argument is named cat
>>myfun(1,2+cat)
The second argument is named
```

函数 inputname 用来显示警告或错误信息中的参数名非常有用。

例 5.3 选择性参数的应用

通过创建函数把直角坐标值（x，y）转化相应的极坐标值，我们向大家说选择性参数的应用。这个函数支持两个输入参数，x 和 y。但是，如果支持只有一个参数的情况，那么函数就假设 y 值为 0，并使用它进行运算。函数在一般情况下输出量为模与相角（单位为度）。但只有一个输出参数只有一个时，它只返回模。函数如下所示。

```
function [mag, angle] = polar_value(x, y)
%     POLAR_VALUE Converts(x, y) to (r, theta)
%     Punction POLAR_VALUE converts an input(x,y)
%     va1ue into (r, theta), with theta in degrees.
%     It illustrates the use of optional arguments.
%     Define variables:
%      angle                  --Angle in degrees
%      msg                    --Error message
%      mag                    --Magnitude
%      x                      --Input x value
%      y                      --Input y value(optional)
%      Record Of revisions:
%      Date       Programmer          Description of change
% ======== ============== ========================
% 12/16/98 S.J.Chapman           Original code
% Check for a legal number of input arquments
msg = nargchk(1,2,nargin);
error(msg);
% If the y argument is missing, set it to 0.
if nargin < 2
      y = 0;
end
% Check for (0,0) input argument, and print out
% a warning message.
if x == 0 & y == 0
      msg = 'Both x and y are zero: angle is meaningless!';
      warning(msg);
end
% Now calculate the magnitude
mag = sqrt(x .^2 + y .^2);
% If the second output argument is present,calculate
% angle in degrees
if nargout == 2
      angle = atan2(y,x) * 180/pi;
end
```

我们通过在命令窗口反复调用这个函数来检测它。首先，我们用过多或过少的参数来调用这个函数。

```
>> [mag angle]=polar_value
??? Error using ==> polar_value
Not enough input arguments.

>> [mag angle]=polar_value(1,-1,1)
??? Error using ==> polar_value
Too many input arguments.
```

在两种情况下均产生了相应的错误信息。我们将用一个参数或两个参数调用这个函数。

```
>> [mag angle]=polar_value(1)
mag =
      1
```

```
angle =
      0
>> [mag angle]=polar_value(1,-1)
mag =
    1.4142
angle =
   -45
```

在这两种情况下均产生了正确的结果。我们调用这个函数使之输出有一个或两个参数。

```
>> mag = polar_value(1,-1)
mag =
    1.4142
>> [mag angle]=polar_value(1,-1)
mag =
    1.4142
angle =
   -45
```

这个函数提供了正确的结果。最后当 x=0，y=0 时，调用这个函数。

```
>> [mag angle] = polar_value(0,0)
Warning: Both x and y are zero: angle is meaningless!
> In polar_value at 27
mag =
      0
angle =
      0
```

在这种情况下，函数显示了警告信息，但执行继续。

注意一个 **MATLAB** 函数将会被声明有多个输出函数，超出了实际所需要的，这是一种错误。事实上，函数没有必要调用函数 nargout 来决定是否有一个输出参数存在。例如，考虑下面的函数。

```
function [z1, z2] = junk(x, y)
z1 = x +    y;
z2 = x - y;
```

这个函数输出可以有一个或两个输出参数。

```
>> a = junk(2,1)
a =
      3
>> [a b] = junk(2,1)
a =
      3
b =
      1
```

在一个函数中检查 nargout 的原因是为了防止无用的工作。如果我们找不到输出结果，为什么不在第一位置计算出来？程序员可以不必为无用的运算耐恼，也能加速程序的运算。

测试 5.1

本测试提供了一个快速的检查方式，看你是否掌握了 5.1 到 5.3 的基本内容。如果你对本测试有疑问，你可以重读 5.1 到 5.3，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. 脚本文件与函数的区别是什么？
2. 自定义函数的 help 命令是如何工作的？
3. 函数中的 H1 注释行有什么重要性？

4. 什么是按值传递机制？它对结构化编程有什么好处。
5. 如何使 **MATLAB** 函数带有选择性参数。

第 6，7 题中，请你确定函数的调用是否正确。如果它是错误的，指出错误所在。

6.

```
out = test1(6);

function res = test1(x, y)
res = sqrt(x .^2 + y .^2);
```

7.

```
out = test2(12);

function res = test2(x, y)
error (nargchk(1,2,nargin));
if nargin == 2
    res = sqrt(x .^2 + y .^2);
else
    res = x;
end
```

5.4 用全局内存分享数据

我们已经看到了，函数与程序之间交换数据是通过参数列表来完成的。当一个函数被调用时，每一个实参都会被复制，而这个复制量将会在函数中用到。

对于参数列表还有一些补充，**MATLAB** 函数与每一个参数或基本工作区通过全局内存交换数据。全局内存是指内存的一种特殊类型，它能够被所有的工作区访问。如果一个变量在函数中被声明全局变量，那

么它将占用的是全局内存，而不是本地工作区。如果相同的变量在另一个函数中被声明为全局变量，那么这个变量所占有内存区域就是第一个函数中的相同变量。声明有全局变量的脚本文件或函数将有办法访问相同的值，所以全局变量为函数之间分享数据提供了一个方法。

全局变量的声明要用到 global 主语句，基本形式如下

```
global var1 var2 var3 ...
```

其中 var1，var2，var3 等等是用全局内存的变量。为了方便，全局变量将在函数开头被声明，但是实际上没有这个必要。

好的编程习惯

最是把全局变量声明在函数的开头，这样可以区别于本地变量。

每一个全局变量在函数第一次使用之前必须声明如果在本地工作区中已经被创建，那么声明为再次声明全局变量将会产生错误。为了避免这种错误，在函数中的初始注释行之后和第一个可执行性语句之前声明全局变量。

好的编程习惯

在函数中的初始注释行之后和第一个可执行性语句之前声明全局变量

全局变量尤其适用于在许多函数分享大容量数据，这样全部的数据在每一次被函数调用时就不必再复制了，用全局变量在函数之间交换数据的不利一面为函数只能为特定的数据工作。通过输入数据参数交换数据的函数能用不同的参数调用它，而用全局变量进行数据交换的函数必须进行修改，以许它和不同的数据进行工作。

好的编程习惯

在一个程序，你能利用全局内存，在函数之间对大规模数据进行交换。

例 5.4

随机数发生器对真实世界进行精确度量是十分重要的。对于每一个测量值来说，都有一定的测量噪声（误差）。对于系统的设计来说，这个情况必须要重要考虑，例如，真实世界中机械装置（飞机等）的运转。好的工程设计必须把他的测量误差控制在一定的范围之内，不能因误差导致系统的不稳定。

在系统建立之前，许多的工程设计的检测是通过系统操作的模拟（simulation）来完成的。模拟包括系统动作的数学模型和符合这个模型的输入数据。如果这个模型对**模拟输入数据**反应正确，那么我们就能合理的证明，真实世界中的系统对真实世界中输入值有正确的反应。

提供给数学模型的**模拟输入数据**必须带有**模拟测量噪声**。模拟测量噪声是指加入理想输入值中的一系列随机数。模拟噪声一般由**随机数发生器**产生。

随机数发生器是一个函数，当它每一次被调用时，将会返回一个不同的随机出现的数。事实上，这些数是由一个**确定性算法**产生的，它们只是表现为随机。但是，如果产生它们的算法足够复杂，那么应用于模拟中的这些数就足够地随机。

下面是一个简单随机数发生器的算法。它是利用大数求余的不可预知性。考虑下面的等式。

$$n_{i+1} = mod(8121n_i + 28411, 134456) \qquad (5.6)$$

假设 n_i 为非负整数，那么由于求余函数的关系，n_{i+1} 只能在 0 到 13445 之间的整数中进行取值。重复以上过程，得到的结果永远是在区间[0, 134455]中。如果我们事先不知道 8121，28411 和 134456 这三个数你很可能猜测这个顺序是由 n 值产生的。进一步说，它说明，所有在 0 到 13445 之间的整数出现的次序是等可能性。由于这些属性，等式（5.6）可以当一个简单的随机数发生器的基础。

现在我们用公式（5.6）设计一个随机数发生器，它的输出是一个实数，其取值范围这[0.0, 1.0]。

答案

我们要编写一个函数，在每一次调用时，它能产生 0≤ran<1.0 的随机数。随机数的产生将依赖于下面的公式。

$$ran_i = \frac{n_i}{1334456} \qquad (5.7)$$

通过公式 5.6n_i，在 0 到 134455 之间进行取值。公式 5.6，5.7 中产生的随机数的顺序取决于 n_0 的初始值(种子，seed)。我们要为用户提供一种途径，让它用于指定 n_0，这样每次运行这个函数得到的随机数顺序都是不一样的。

1.陈述问题

编写一个函数 random0，使之产生一个数组，数组中包括一个或多个随机数，它的取值范围是 0≤ran<1.0，它的顺序由公式 5.6 和 5.7 指定。函数应当有一个或多个输入参数(n 和 m)，用来指定返回数组的大小。如果它有一个参数，函数将产生一个 n 阶方阵;如果有两个参数，函数将会产生一个 $n×m$ 的数组。种子 n_0 的初始值将会由函数 seed 指定。

2.定义输入量和输出量

在这个问题中共有两个函数:seed 和 random0。函数 seed 的输入是一个整数。这个函数没有输出。random0 的输入量是一个或两个整数。如果它有一个参数，函数将产生一个 n 阶方阵；如果有两个参数，函数将会产生一个 $n×m$ 的数组。这个函数的输出是由在 0.0 和 1.0 之间的随机数组成的数组

3.定义算法

函数 random0 的伪代码如下:

```
function ran = random0 (n, m)
Check for valid arguments
Set m ← n if not supplied
Create output array with "zeros" function
```

```
for ii = 1:number of rows
    for jj = 1: number of columns
        ISEED ← mod (8121 * ISEED + 28441, 134456)
        ran(ii,jj) ← ISEED /134456
    end
end
```

其中，ISEED 是全局变量，所以它能被所有的函数调用。函数 seed 的伪代码如下:

```
function seed ( new_seed)
new_seed ← round( new_seed)
ISEED ← abs( new_seed)
```

当用户输入不是整数，round 函数会对其进行四舍五入。如果输入的是一个负数，abs 将会把他取正。用户事先不需知道只有正整数才是合法的种子。

ISEED 是全局变量，所以它能被所有的函数调用。

4.把算法转化为 **MATLAB** 语句

函数 random0 的代码如下:

```
function ran = random0(n,m)
%RANDOM0 Generate uniform random numbers in [0,1)
% Function RANDOM0 generates an array of uniform
% random numbers in the range [0,1). The usage
% is:
%
% random0(n)          --Generate an n x n array
% random0(n,m)        --Generate an n x m array
% Define variables:
% ii                  --Index variable
% ISEED               --Random number seed (global)
% jj                  --Index variable
% m                   --Number of columns
% msg                 --Error message
% n                   --Number of rows
% ran                 --Output array
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/16/98 S. J. Chapman Original code
% Declare globl values
global ISEED          % Seed for random number generator
% Check for a legal number of input arguments.
msg = nargchk(1,2,nargin);
error(msg);
% If the m argument is missing, set it to n.
if nargin < 2
    m = n;
end
% Initialize the output array
ran = zeros(n,m);
% Now calculate random values
for ii = 1:n
    for jj = 1:m
        ISEED = mod(8121*ISEED + 28411, 134456 );
        ran(ii,jj) = ISEED / 134456;
    end
end
```

函数 seed 的代码如下:

```
function seed(new_seed)
```

```
%SEED Set new seed for function RANDOM0
% Function SEED sets a new seed for function
% RANDOM0. The new seed should be a positive
% integer.
% Define variables:
% ISEED                --Random number seed (global)
% new_seed             --New seed
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/16/98 S. J. Chapman Original code
% Declare globl values
global ISEED % Seed for random number generator
% Check for a legal number of input arguments.
msg = nargchk(1,1,nargin);
error(msg);
% Save seed
new_seed = round(new_seed);
ISEED = abs(new_seed);
```

5.检测产生的 **MATLAB** 程序

如是程序产生的这些数是真正的取值范围在 0≤ran<1.0 的等可能性随机数，那么它们的平均数应接近 0.5，它们的标准差应接近 $\frac{1}{\sqrt{12}}$。

进一步说，如果一个如果把区间[0, 1)分许多相同长度的子区间。那么落在每一个子区间的随机数的数目应当是相同的。我们可以利用柱状统计图来统计落于每一个子区间的随机数的数目。**MATLAB** 函数 hist 能够读取输入数据并能创建出相应的柱状图，所以我们将利用它随机数的等可能性。

1.调用函数 seed，把 new_seed 设置为 1024

2.调用 random0(4)，观察得到的结果

3.再次调用 rnadom0(4)，证明每次产生的数不相同

4.重新调用函数 seed，把 new_seed 设置为 1024

5.再次调用 rnadom0(4)，观察与 2 得到的结果是否相同

6.调用 random0(2, 3)证明函数可以输入两个参数

7.调用 random0(1, 20000)并计算产生的数组的平均数和标准差。看得到结果是否分别与 0.5，$\frac{1}{\sqrt{12}}$ 接近。

```
>> seed(1024)
>> random0(4)
ans =
    0.0598    1.0000    0.0905    0.2060
    0.2620    0.6432    0.6325    0.8392
    0.6278    0.5463    0.7551    0.4554
    0.3177    0.9105    0.1289    0.6230
>> random0(4)
ans =
    0.2266    0.3858    0.5876    0.7880
    0.8415    0.9287    0.9855    0.1314
    0.0982    0.6585    0.0543    0.4256
    0.2387    0.7153    0.2606    0.8922
>> seed(1024)
>> random0(4)
ans =
    0.0598    1.0000    0.0905    0.2060
    0.2620    0.6432    0.6325    0.8392
```

```
        0.6278    0.5463    0.7551    0.4554
        0.3177    0.9105    0.1289    0.6230
>> random0(2,3)
ans =
        0.2266    0.3858    0.5876
        0.7880    0.8415    0.9287
>> arr = random0(1,20000);
>> mean(arr)
ans =
        0.5020
>> std(arr)
ans =
        0.2881
>> hist(arr,10);
>> title('\bf Historygram of the Output of random0');
>> xlabel('Bin')
>> ylabel('Count')
```

检测的结果是合理的，这些数据的平均值为 0.5020，理论值为 0.5，实际标准差为 0.2881，理论值为 0.2887 接近。柱状图如图 5.5 所示。

在 **MATLAB** 中，有两个产生随机数的内建函数。它们是

rand　　　　　用于产生等可能的随机数
randn　　　　 用于产生普通的随机数

这两个函数要远比我们创建这个随机数发生器要快得多，产生的随机数也多得多。如果你需要在你的程序中创建一些随机数，可调用它们。

调用函数 rand 和 randn 的形式如下

rand　　　　　产生一个随机数
rand(n)　　　 产生一个 n×n 的随机数数组
rand(n, m)　　产生一个 n×m 的随机数数组

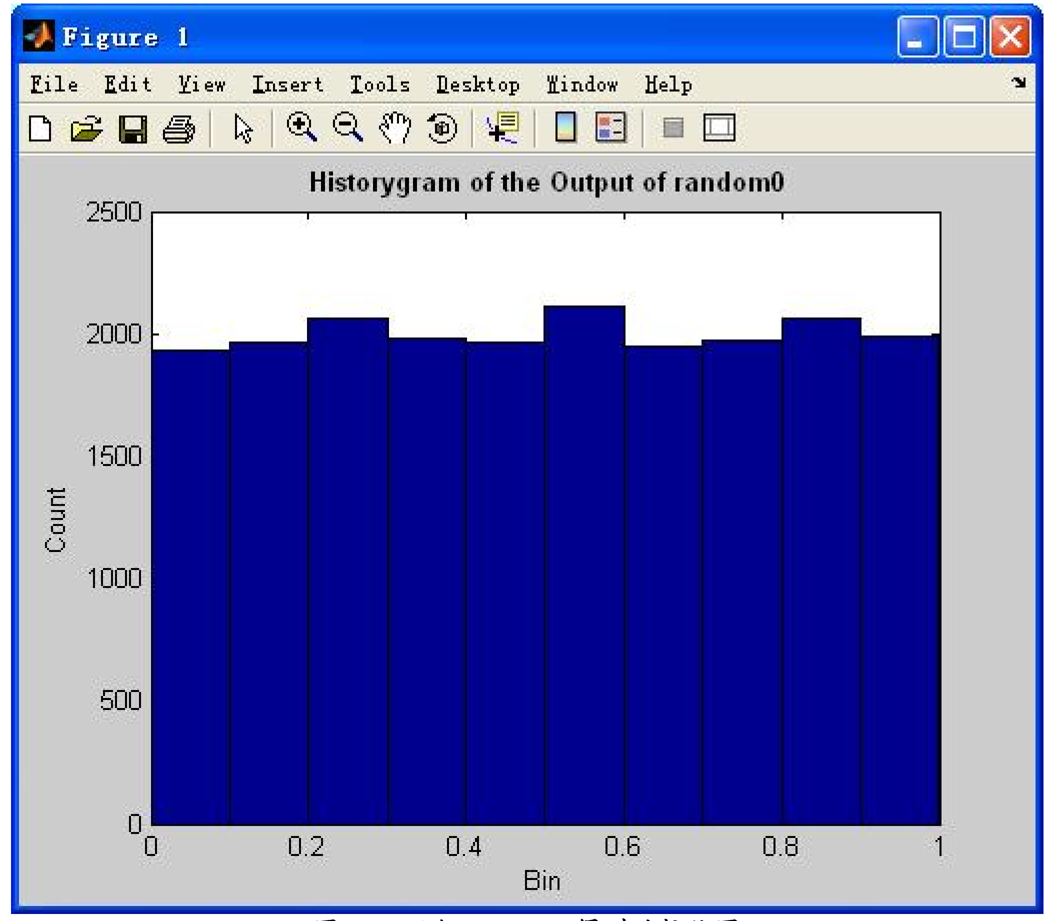


图 5.5 函数 random0 得到的柱状图

5.5 在函数调用两次之间本地数据的存储

当一个函数执行结束，由这个函数创建的特定的工作区将会被破坏，所以在这个函数中的所有本地变量将会消失。当这个函数下一次被调用的时侯，一个新的工作区将会被创建，而且所有本地变量的值都返回为默认。这种特性是我们所期望的，因为只有这样 **MATLAB** 函数才能被重复调用而不受上一次影响。

但在有些情况下，多次调用一个函数，存储一些本地变量的信息还是有用的。例如，我们想创建一个计数器，对函数调用的次数进行计数。如果每一次函数结束执行，计算器就会被破坏，那么计数不超过 1。

从 **MATLAB**5.1 开始，**MATLAB** 中就有了一个特殊的机制。这种机制允许多次调用一个函数时，保存本地变量。**持久内存**(persistent memory)是内存的一种类型，在函数上一次调用之后，这一步调用之前，本地变量被保存在持久内存，值不变。

持久变量应用语句声明。它的形式如下:

```
persistent var1 var2 var3 ...
```

var1，var2，var3...是存储于持久内存中的变量。

好的编程习惯
在两次函数调用之间有持久内存保存本地数据。

例 5.5 运行平均数

当我们键入一些变量总想得到他的统计量。**MATLAB** 内建函数 mean 和 std 就是进行统

计数据运算的。我们对一系列的数利用这两个函数进行运算后，再键入一个新数，重新计算。这时我们就可以利用持久内存提高运算的效率。

算术平均数的定义如下:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5.8}$$

其中 x_i 是 N 样品中的第 i 个样品。标准差的定义如下:

$$s = \sqrt{\frac{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}{N(N-1_}} \tag{5.9}$$

标准差则体现随机变量取值与其期望值的偏差。标准差的值较大，则表明该随机变量的取值与其期望值的偏差较大，反之，则表明此偏差较小。如果我们能够记录下样本的个数 N，样本的和 Σx_i 以及样本的平方和 Σx_i^2，我们在任何的时侯就可以通过公式（5.8）和（5.9）计算出它的平均数和标准差。编写一个程序，计算当前输入数据的当前输入数据的平均数和标准差。

答案

函数必须能够每接受一次输入值并记录下对应的 N，Σx_i 和 Σx_i^2，用于计算当前的平均数和标准差。N，Σx_i 和 Σx_i^2 必须存储在持久内存中，这样在两次调用之间，它不会消失。最后函数必须有一种机制，把运行的和清零。

1.陈述问题

编写一个程序，计算当前输入数据的当前输入数据的平均数和标准差。函数必须有一种机制，把运行的和清零。

2.定义输入输出值

这个函数需要两种类型的输入量

(1)字符型变量“reset”用于 N，Σx_i 和 Σx_i^2 的清零

(2)每一次调用函数用于参与运算的数字类型数据这个函数的输出量为到目前为止函数所接受的所有数字数据的数学期望和平方和。

(3)设计算法这个函数被分为四大步。

```
Check for a legal number of arguments
Check for a 'reset', and reset sums if present
Otherwise, add current value to running sums
Calculate and return running average and std dev
        if enough data is available. Return zeros if not enough data is available.
```

这些步骤的伪代码为

```
Check for a legal number of arguments
if x == 'reset'
  n ← 0
  sum_x ← 0
  sum_x2 ← 0
else
  n ← n+1
  sum_x ← sum_x + x
  sum_x2 ← sum_x2 + x^2
end

% Calculate ave and sd
if n == 0
  ave ← 0
  std ← 0
elseif n == 1
  ave ← sum_x
```

```
  std ← 0
else
  ave ← sum_x / n
  std ← sqrt((n*sum_x2 – sum_x^2) / (n * (n-1)))
end
```

4.把算法转化为 **MATLAB** 代码

```
function [ave, std] = runstats(x)
%RUNSTATS Generate running ave / std deviation
% Function RUNSTATS generates a running average
% and standard deviation of a data set. The
% values x must be passed to this function one
% at a time. A call to RUNSTATS with the argument
% 'reset' will reset tue running sums.
% Define variables:

% ave          --Running average
% msg          --Error message
% n            --Number of data values
% std          --Running standard deviation
% sum_x        --Running sum of data values
% sum_x2       --Running sum of data values squared
% x            --Input value
% Record of revisions:
% Date      Programmer      Description of change
% ====      ==========      =====================
% 12/16/98  S. J. Chapman   Original code
% Declare persistent values
persistent n            % Number of input values
persistent sum_x        % Running sum of values
persistent sum_x2       % Running sum of values squared
% Check for a legal number of input arguments.
msg = nargchk(1,1,nargin);
error(msg);

% If the argument is 'reset', reset the running sums.
if x == 'reset'
    n = 0;
    sum_x = 0;
    sum_x2 = 0;
else
    n = n + 1;
    sum_x = sum_x + x;
    sum_x2 = sum_x2 + x^2;
end

% Calculate ave and sd
if n == 0
    ave = 0;
    std = 0;
elseif n == 1
    ave = sum_x;
    std = 0;
else
    ave = sum_x / n;
    std = sqrt((n*sum_x2 - sum_x^2) / (n*(n - 1)));
```

```
end
```

5.检测程序

为了检测这个函数，我们必须创建一个用于复位 runstats 的脚本文件：读取输入数据，调用 rnnstats 函数，并显示出相应的统计量。

一个合适的脚本文件显示如下：

```
% Script file: test_runstats.m
%
% Purpose:
% To read in an input data set andn calculate the
% running statistics on the data set as the values
% are read in. The running stats will be written
% to the Command window.
%
% Record of revisions:
% Date         Programmer        Description of change
% ====         ==========        =====================
% 12/16/98     S. J. Chapman     Original code
%
% Define variables:
% array                    --Input data array
% ave                      --Running average
% std                      --Running standard deviation
% ii                       --Index variable
% nvals                    --Number of input values
% std                      --Running standard deviation

% First reset running sums
[ave std] = runstats('reset');
% Prompt for the number of values in the data set
nvals = input('Enter number of values in data set: ');
% Get input values
for ii = 1:nvals
    % Prompt for next value
    string = ['Enter value ' int2str(ii) ': '];
    x = input(string);
    % Get running statistics
    [ave std] = runstats(x);
    % Display running statistics
    fprintf('Average = %8.4f; Std dev = %8.4f\n',ave, std);
end
```

为了检测函数，通过手动计算 5 个数得到相应的统计量，并与程序得到的结果进行比较。如果这个 5 个数分别为：

```
3.0, 2.0, 3.0, 4.0, 2.8
```

那么手动运算的结果为

Value	n	Σx	Σx^2	Average	Std_dev
3.0	1	3.0	9.0	3.00	0.000
2.0	2	5.0	13.0	2.50	0.707
3.0	3	8.0	22.0	2.67	0.577
4.0	4	12.0	38.0	3.00	0.816
2.8	5	14.8	45.84	2.96	0.713

程序得到的结果为

```
>> test_runstats
Enter number of values in data set: 5
Enter value 1: 3
Average =     3.0000; Std dev =     0.0000
```

```
Enter value 2: 2
Average =     2.5000; Std dev =     0.7071
Enter value 3: 3
Average =     2.6667; Std dev =     0.5774
Enter value 4: 4
Average =     3.0000; Std dev =     0.8165
Enter value 5: 2.8
Average =     2.9600; Std dev =     0.7127
```

这个运算结果与上面的手动计算相符。

5.6 函数的函数(function functions)，

函数的函数(function functions)是指函数的输入参数中含有其他的函数，传递给函数的函数的变量名一般情况应用于这个函数执行的过程中。

例如，**MATLAB** 中有一个函数的函数叫做 fzero。这个函数用于找到传递给它的函数值为 0 时的自变量。例如，语句 fzero('cos', (0, pi))，它能确定 cos 函数在区间[0, π]中何时为 0。语句 fzero('exp(x)-2', [0 1])在区间[0, 1]中何时为 0。当这些语句被执行时，将产生如下的结果：

```
>> fzero('cos',[0 pi])
ans =
      1.5708
>> fzero('exp(x)-2',[0 1])
ans =
      0.6931
```

函数的函数操作的关键字有两个专门的 maltab 函数，eval 和 feval。。函数 eval 对一个字符串进行求值，就如它在命令窗口中已经键入了一样。函数 feval 用一个特定的输入值对命名的函数进行求值。函数 eval 的形式如下：

```
eval(string)
```

例如，语句 x = eval('sin(pi/4)')产生的结果如下：

```
>> x = eval('sin(pi/4)')
x =
      0.7071
```

下面是一个例子，构建一个字符串，并用 eval 函数对其进行求值

```
x = 1;
str = ['exp(' num2str(x) ')-1'];
res = eval(str);
```

在这种情况下，变量 str 的内容为 exp(1)-1，所以 eval 产生的结果为 1.7183。

函数 feval 对在 M 文件进行定义的命名函数进行求值，要求有指定的输入值。函数 feaval 的基本形式如下

feval(fun, value).

例如，语句 x=feval('sin'，pi/4)产生的结果如下

```
>> x = feval('sin',pi/4)
x =
      0.7071
```

更多的函数的函数将会在表 5.1 中列出。在命令窗中键入 help 函数名，了解他们的用途。例 5.6

表 5.1 常见的函数的函数

fminbnd	求函数的最小值，这函数只有一个自变量
fzero	找出函数为 0 时的自变量的值
quad	在数学上组合一个函数
ezplot	简单易用的函数画图
fplot	通过函数名画出这个函数的图象

创建一个函数的函数，它能够画出所有只有一个自变量的 **MATLAB** 函数的图象，自变量的范围是用户指定的始值和终值。

答案

这个函数有两个输入参数，第一个是要画的函数的函数名，第二个是两元素向量，它指明了画图的取值范围。

1.陈述问题

创建一个函数的函数，它能够画出所有只有一个自变量的 **MATLAB** 函数的图象，自变量的范围由用户指定。

2.定义输入输出函数的输入有两个

(1)包含有函数名的字符串

(2)包含有起始值和终值的 2 元素向量函数的输出是所要画的图象

3.设计算法这个函数可以分为 4 大步：

Check for a legal number of arguments
Check that the second argument has two elements
Calculate the value of the function between the start and stop points
Plot and label the function

第三四大步的伪代码如下

```
n_steps ← 100
step_size ← (xlim(2) – xlim(1)) / nsteps
x ← xlim(1):step_size:xlim(2)
y ← feval(fun, x)
plot(x, y)
title(['bf Plot of function ' fun ' (x)'])
xlabel('\bfx;)
ylabel(['bf ' fun ' (x)'])
```

4.把算法转化为 **MATLAB** 语句

```
function quickplot(fun,xlim)
%QUICKPLOT Generate quick plot of a function
% Function QUICKPLOT generates a quick plot
% of a function contained in a external mfile,
% between user-specified x limits.
% Define variables:
% fun          --Function to plot
% msg          --Error message
% n_steps      --Number of steps to plot
% step_size    --Step size
% x            --X-values to plot
% y            --Y-values to plot
% xlim         --Plot x limits
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/17/98 S. J. Chapman Original code
% Check for a legal number of input arguments.
msg = nargchk(2,2,nargin);
error(msg);
% Check the second argument to see if it has two
% elements. Note that this double test allows the
% argument to be either a row or a column vector.
if ( size(xlim,1) == 1 & size(xlim,2) == 2 ) | ...
( size(xlim,1) == 2 & size(xlim,2) == 1 )
    % Ok          --continue processing.
    n_steps = 100;
    step_size = (xlim(2) - xlim(1)) / n_steps;
    x = xlim(1):step_size:xlim(2);
```

```
        y = feval(fun,x);
        plot(x,y);
        title(['\bfPlot of function ' fun '(x)']);
        xlabel('\bfx');
        ylabel(['\bf' fun '(x)']);
    else
        % Else wrong number of elements in xlim.
        error('Incorrect number of elements in xlim.');
    end
```

5。检测程序为了检测这个程序，我们应当用正确或错误的输入输出参数来调用这个程序。证明它能区分正确和错误的输入参数。结果如下

```
>> quickplot('sin')
??? Error using ==> quickplot
Not enough input arguments.

>> quickplot('sin', [-2*pi 2*pi], 3)
??? Error using ==> quickplot
Too many input arguments.

>> quickplot('sin', -2*pi)
??? Error using ==> quickplot
Incorrect number of elements in xlim.

>> quickplot('sin', [-2*pi 2*pi])
```

最后一次被调用的结果是正确的，它的图象如图 5.6 所示。

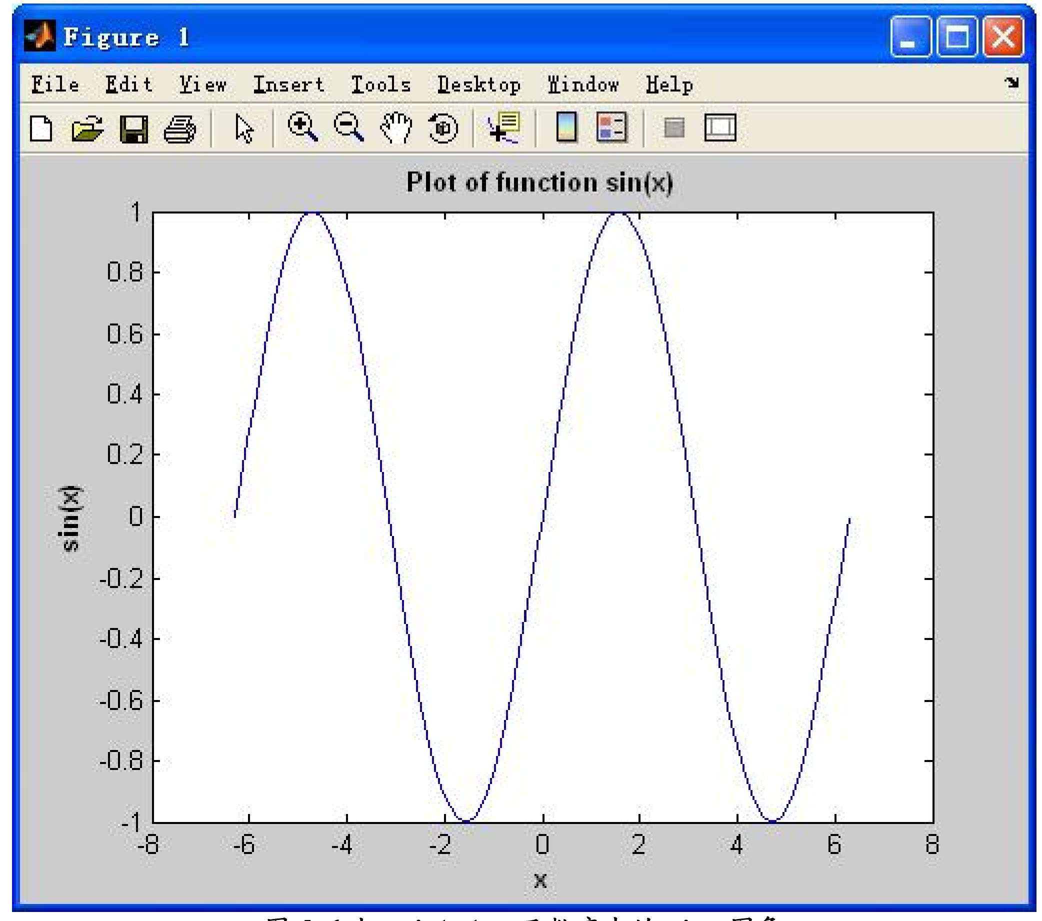


图 5.6 由 quickplot 函数产生的 sinx 图象

5.7 子函数和私有函数

在一个单个的文件中我们可以创建多个函数。如果超过 1 个的函数出现在一个文件中，那么最上面的那个函数为普通函数，下面的函数称为**子函数**或**中间函数**。子函数看起来和普通函数一样，但是只能被同一文件中的函数调用。

下面的例子定义了一个函数 mystats 和两个子函数 mean 和 median。函数 mystats 能被其他的 maltab 函数调用，但是子函数 mean 和 median 只能同一文件中的其他函数调用。

```
function [avg, med] = mystats(u)
%MYSTATS Find mean and median with internal functions.
%     Function MYSTATS calculates the average and median
%     of a data set using subfunctions.

n = length(u);
avg = mean(u, n);
med = median (u, n);

function a = mean(v, n)
%     Subfunction to calculate average.
a = sum(v) / n;

function m = median(v, n)
%     Subfunction to calculate median

w = sort(v);
if rem(n, 2) ==1
       m = w((n+1)/2);
else
       m = (w(n/2) + w(n/2 + 1))/2;
end
```

私有函数是指属于以 private 为名字的子目录中的函数。这些函数只有在父目录中才是可见的。例如，假设在 **MATLAB** 搜索路径中有一 testing 目录，在 testing 目录中，又有一个 private 子目录。private 中的函数只能由 testing 中的函数调用。因为对于父目录以外目标私有函数是不可见的，所以它能用其他目录中的函数重名。有了这种特性，如果你要创建自己的函数，则不必要考虑与其他目录重名。因为 **MATLAB** 先对私有函数查找，然后再对标准的 M 文件函数进行查找，所以它将首先找到私有函数 test.m，再找到非私有 M 文件 test.m。

在包含有你的函数的目录中，我们可以很创建你的私有目录。不要在你的搜索路径中放置你的私有目录。

在一个 M 文件中，调用一个函数，**MATLAB** 先检查看他是否是一个子函数。如果它不是那就检查它是不是一个私有函数。如果也不是私有函数，**MATLAB** 就会检它在不在标搜索路径中。

如果你有特殊的目的，**MATLAB** 函数只能由其他的函数调用，而绝不能由使用者调用并考虑用子函数或私有函数来隐藏它们。隐藏这些函数防止了它们偶然的使用，也能防止与其他公共函数重名时发生的冲突。

好的编程习惯

用子函数或私有函数来隐藏特殊目的的函数，这些隐藏的函数只能被其他函数调用。隐藏这些函数防止了它们偶然的使用，也能防止与其他公共函数重名时发生的冲突。

5.8 总结

在第五章中，我们向大家介绍了用户自定义函数。函数是 M 文件的一种特殊类型，它通过输入参数接受数据，通过输出参数返回结果。每一个函数都有其独立的工作区。

MATLAB 通过按值传递机制将参数传递给函数，这意味着 **MATLAB** 把每一个参数复制，并把这个拷贝传递给函数。这个复制是非常重要的，因为函数可以自由的修改输入参数，而不会影响到程序中的实参。

MATLAB 函数支持改变输入输出参数的个数。函数 nargin 可以报告函数在调用过程中所需的实参个数。函数 nargout 则可以报告输出参数的个数。

把数据存于全局内存中，可以实现 **MATLAB** 函数之间数据的共享。全局变量的声明要用到 global 语句。全局变量可由所有声明它的所有函数共享。为了方便，全局变量应在 M 文件的开头声明。

两次调用同一函数之间，中间数据可以存储在持久内存。持久变量可能用 persistent 语句声明。

函数的函数是指函数的输入参数中含有其他的函数，传递给函数的函数的变量名一般情况应用于这个函数执行的过程中。

子函数是在一个单独文件中的附加函数，它只能被同一文件中的其他函数访问。私有函数是在 private 子目录中的函数，它们只能被父目录中的函数访问。子函数和私有函数主要用于限制 **MATLAB** 函数的访问。

5.8.1 好的编程习惯的总结

1.把大的程序分解小的，易于理解的函数

2.在 M 文件的开头声明全局变量。以区分本地变量

3.在函数中的初始注释行之后和第一个可执行性语句之前声明全局变量

4.全局变量适用大规模数据的传输

5.在两次函数调用之间有持久内存保存本地数据。

6.用子函数或私有函数来隐藏特殊目的的函数，这些隐藏的函数只能被其他函数调用。隐藏这些函数防止了它们偶然的使用，也能防止与其他公共函数重名时发生的冲突。

5.8.1 MATLAB 总结

下面是对 **MATLAB** 函数和命令的总结，并带有简短的描述。

nargin	这个函数返回调用这个函数时所需要的实际输入参数的个数
nargout	这个函数返回调用这个函数时所需要的实际输出参数的个数
nargchk	如要一个函数调用被调用时参数过多或过少，那么 nargchk 函数将返回一个标准错误信息
error	显示错误信息，并中止函数以免它产生这个错误。如果参数错误是致命的，这个函数将会被调用。
warning	显示警告信息并继续执行函数，如果参数错误不是致命的，执行还能继续，则这个将会被调用。
inputname	这个函数将会返回对于特定参数个数的实际变量名。
rand	产生一个随机数
rand(n)	产生一个 n×n 的随机数数组
rand(n,m)	产生一个 n×m 的随机数数组
rand	用于产生等可能的随机数
randn	用于产生普通的随机数

5.9 练习

5.1 函数与脚本文件之前的区别是什么？

5.2 当一个函数被调用，数据是是怎样从调用者传递到函数的。函数是怎样把结果返回给调用者?

5.3 在 **MATLAB** 中应用按图传递机制的优点与缺点?

5.4 修改本章中的选择性排序函数，让他能够接受第二个选择性参数。"up"或"down"。当参数为"up"时，数据按升序排列，当参数为"down"，数据按降序排列。如果输入只有一个，我们默认按降序排列。

5.5 编写一个函数，利用函数 random0 在[-1。0，1。0]产生一个随机数。使 random0 函数成为你新的函数的子函数。

5.6 编写一个函数，利用函数 random0 在[low,high]产生一个随机数，其中 low，high 分别代表输入参数。把 random0 函数成为你新函数的一个私有函数。

5.7 骰子模拟。模拟掷骰子的情况在现实中非常有用。编写一个 maltab 程序模拟掷骰子，每次产生一个 1 到 6 之间随机整数。

5.8 道路交通密度。函数 random0 将在[0.0, 1.0]产生一个等可能性随机数。如果随机事件结果是等可能性，这个函数适合模拟这类随机事件。但是，很多事件的发生都不是等可能性的，那么这个函数不适合模拟这类情况。

例如，一交通工程师研究在一段时间隔 t 内通过某一地点汽车数，发现 k 辆汽车通过一指定地点可能性为

$$P(k,t)=e^{-\lambda t}\frac{(\lambda t)^k}{k!} \quad (t\geq 0, \lambda>0, k=0,1,2,...) \qquad (5.10)$$

它的分布符合**指数分布**。指数分布在科研和工程上有很多的应用。例如，在时间 t 内接打电话的次 k，指定容器内的病毒 k，以及复合系统的出错次数 k 都符合指数分布。

编写一个函数，对任意 k，t 和 λ 求指数分布。通过计算在 1 分钟内通过高速路上指定一点 1，2，3，4，5 辆汽车的概率。并画出相应的图象。已知 λ 为 1.5。

5.9 编写下面三个函数，用以计算数 x 的双曲正弦，双曲余弦和双曲正切用你的函数画出它们所对应的图象。

$$\sinh(x)=\frac{e^x-e^{-x}}{2}, \cosh(x)=\frac{e^x+e^{-x}}{2}, \tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$$

5.10 向量叉乘。编写一个程序计算向量 V_1 和 V_2 的叉乘积。

$$V_1\times V_2=(V_{y1}V_{z2}-V_{y2}V_{z1})i+(V_{z1}V_{x2}-V_{z2}V_{x1})j+(V_{x1}V_{y2}-V_{x2}V_{y1})k$$

其中 $V_1=V_{x1}i+V_{y1}j+V_{z1}k$，$V_2=V_{x2}i+V_{y2}j+V_{z2}k$。注意这个函数返回的一个实数组。用这个函数计 V_1 = [-2, 4, 0.5]和 V_2 = [0.5, 3, 2]的叉乘积。

5.11 带有搬运的排序。对数组 arr1 进行升序排序，与 arr1 中相对应的 arr2 中的元素也要发生改变。对这个种排序，每次 arr1 中的一个元素与另一个元素进行交换，arr2 中对应的元素也要进行相应的交换。当排序结束时 arr1 中的元素按升序排列，arr2 中的元素也会有相应的变化。例如下面两个数组

Element	arr1	arr2
1.	6.	1.
2.	1.	0.
3.	2.	10.

当 arr1 的数组排序结束后，arr2 也要进行相应的变化。两数组为

Element	arr1	arr2
1.	1.	0.
2.	2.	10.
3.	6.	1.

编写一个程序，对第一个实数组进行按降序排列，对第二个数组进行相应交化。用下面两个数组检测你的程序

```
a = [-1,    11,    -6,    17,    -23,    0,    5,    1,    -1];
b = [-31,   102,   36,   -17,     0,    10,   -8,   -1,    -1];
```

5.12 用帮助工作台查找 **MATLAB** 标准函数 sortrows 的信息，运行 sortrows 函数，和前面练习中的排序函数进行比较。为了达到此目的，创建含有 1000×2 元素的数组的两个副本，

数组中是随机数。应用上面的两个函数分别对第一行进行排序，第二行也对应改变。用 tic 和 toc 函数确定每一个排序执行所需要的时间。你编写的函数的运行速度与标准函数的运行速度相比如何？

5.13 图 5.7 显示是漂浮在海洋上两条船。1 号船所在的位置为（x_1，y_1）按 θ_1 方向运行，2 号船所在的位置为（x_2，y_2）按 θ_2 方向运行。假设一个物体与 1 号船的距离 r_1，并产生 φ_1。编写一个程序计算 2 号船到物体的距离 r_2 和夹角 φ_2。

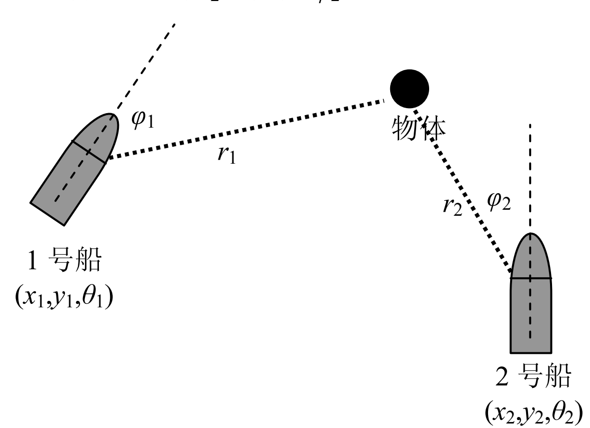


图 5.7 1 号船所在的位置为（x_1，y_1）按 θ_1 方向运行，2 号船所在的位置为（x_2，y_2）按 θ_2 方向运行。

5.14 函数的最大值和最小值。编写一个函数，用于计算任意函数 f(x)在一定区间内的最大值和最小值。所要求最大值和最小值的函数应当用参数的方式传递给你编的函数。这个函数应当有下面的输入参数。

```
first_value        --x 的第一个值
last_value         --x 的最后一个值
num_steps          --x 取值的步长
func       --所要求的值的函数名
```

函数的输出参数应为

```
xmin          --函数 f(x)为最小值时的 x 值
min_value          --函数 f(x)的最小值
xmax          --函数 f(x)为最大值时的 x 值
max_vlaue          --函数 f(x)的最大值
```

确保输入参数的个数有效，你可以通过 help 和 lookfor 命令得到一定的帮助。

5.15 编写一个程序，用来检测上题中产生的函数。这个检测程序把自定义函数 $f(x)=x^3-5x^2+5x-2$ 传递给函数的函数，并在区间[-1,3]内每隔 1/50 取一次值，找出函数的最大值和最小值，并打印出来。

5.16 函数的微分。函数微分的定义如下

$$\frac{d}{dx}f(x)=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x} \tag{5.11}$$

简化后为

$$f'(x_i)=\frac{f(x_{i+1})-f(x_i)}{\Delta x} \tag{5.12}$$

其中 $\Delta x=x_{i+1}-x_i$。

假设一向量 vect 包括一个函数的每隔 dx 的抽样。编写一个程序，通过 5.12 式计算这个向量的微分。这个函数应该能检查 dx 的值，看它是否大于 0，这样可以防止这个函数发生 0 除错误。为了检测你的函数，你应当在脑中产生一个函数，并知道它的微分是什么。与函数产生的结果进行对比。最好的检测函数应是 sin(x)。它的微分为 $\frac{d}{dx}(\sin x)=\cos x$ 。产生一个向量，这个向量包含一百个 sin$x$ 函数值，x 从 0 开始取值，步长 Δx 为 0.05。用你的函数

对这个向量进行微分，然后产生的结果与正确的结果进行对比。你的函数计算得到的结果与正确的微分结果有多接近？

5.17 带有噪声的微分。我们现在编写一个程序，观察输入噪声对数据微分的影响。第一步，产生一个向量，这个向量包含一百个 sinx 函数值，x 从 0 开始取值，步长 Δx 为 0.05。下一步用 random0 函数产生一系列的噪声，并把它加入到上面向量的抽样中(如图图 5.8)。噪声产生的最大偏差不会超过信号幅度的±0.02。现在我们用上题中微分函数对向量进行运算。得到结果和理率值有多相近呢？

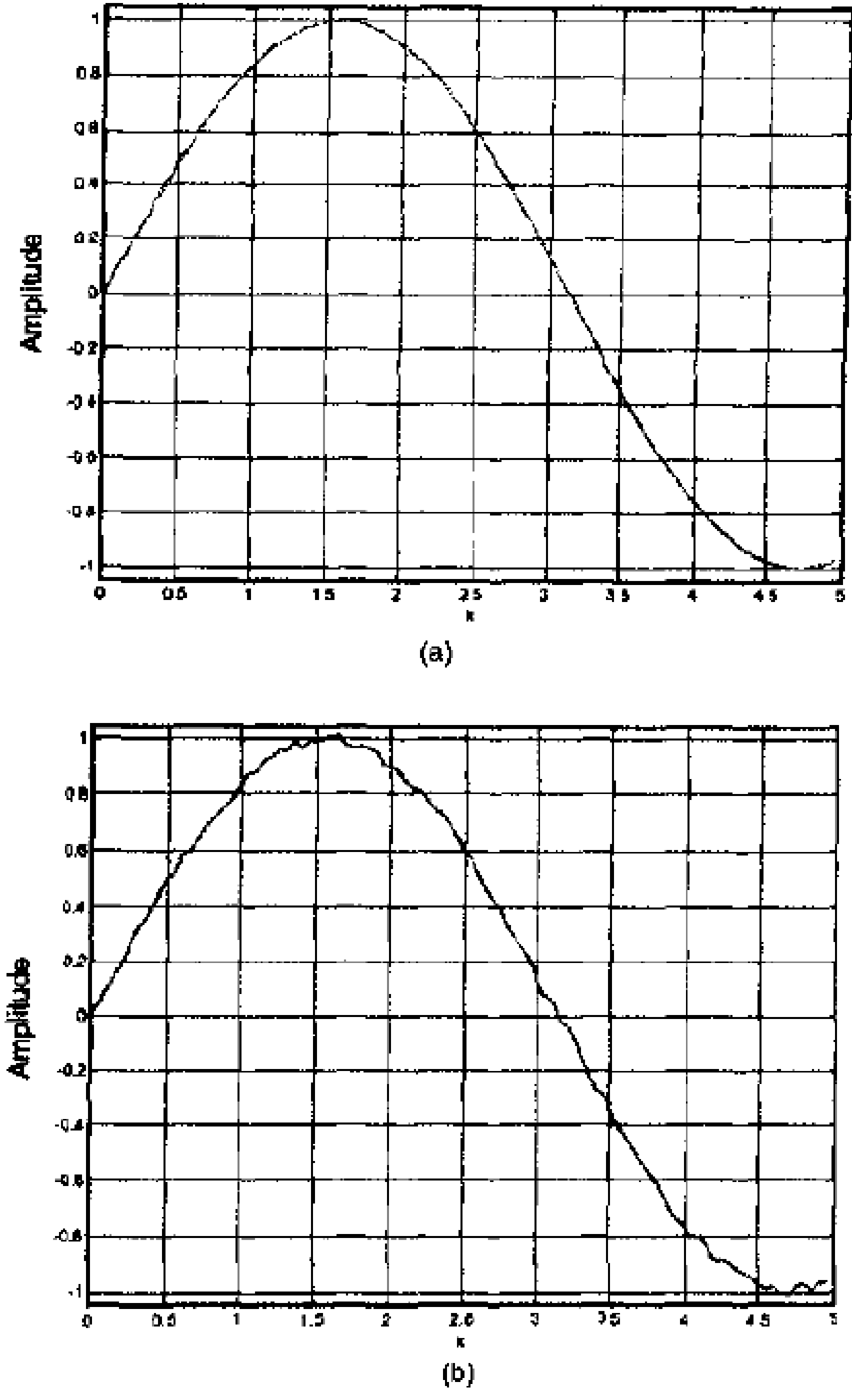


图 5.8 (a)无噪声的 sinx 的图象（b）带有噪声的 sinx 的图象

5.18 线性最小二乘拟合。开发一个函数，用于计算拟合输入数据的最小二乘直线的斜率 m 和截距 b。输入数据点集（x，y）由两个数据传递给函数，数组 x 和 y。用一个检测程序检测你的函数，表 5.2 中有 20 个点输入数据。

表 5.2

No.	x	y	No.	x	y
1	-4.91	-8.18	11	-0.94	0.21
2	-3.84	-7.49	12	0.59	1.73
3	-2.41	-7.11	13	0.69	3.96
4	-2.62	-6.15	14	3.04	4.26
5	-3.78	-5.62	15	1.01	5.75
6	-0.52	-3.30	16	3.60	6.67
7	-1.83	-2.05	17	4.53	7.70
8	-2.01	-2.83	18	5.13	7.31
9	0.28	-1.16	19	4.43	9.05
10	1.08	0.52	20	4.12	10.95

5.19 最小二乘拟合的相关系数

开发一个函数，既可以用于计算拟合输入数据的最小二乘直线的斜率 m 和截距 b。又可以计算拟合的相关系数。输入数据点集（x，y）由两个数据传递给函数，数组 x 和 y。计算 m 和 b 的公式在例 4.7 中已经给出。相关系数 r 的计算公式如下所示

$$r=\frac{n(\sum xy)-(\sum x)(\sum y)}{\sqrt{[(n\sum x^2)-(\sum x)^2][(n\sum y^2)-(\sum y)^2]}} \qquad (5.13)$$

Σx 代表 x 值的和

Σy 代表 y 值的和

Σx^2 代表 x 值的平方和

Σy^2 代表 y 值的平方和

Σxy 对应的 x，y 相乘的和

n 代表拟和中包括的点数

用一个检测程序检测你的函数，输入参数与上题相同。

5.20 生日问题

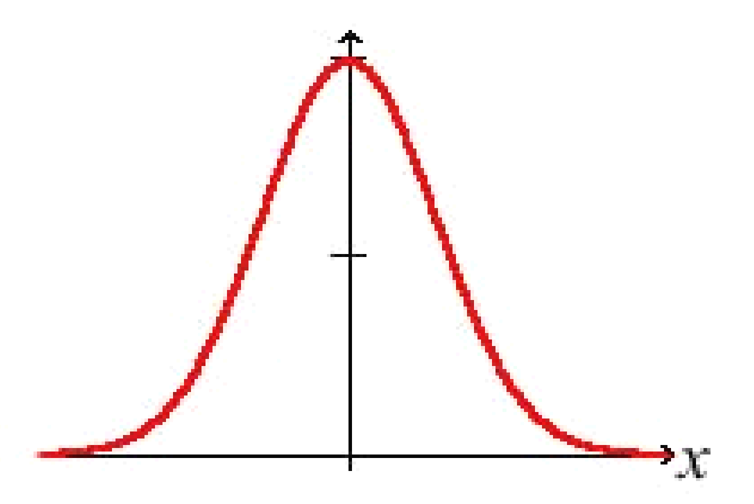
生日问题：如果在一个房间有 n 个人，那么有二个或多个人在同一天过生日的概率为多大？我们可以能过数学建模来解决这一问题。编写一个程序，计算在 n 个人中有二个或多个人在同一天过生日的概率，n 为输入参数。编写一个程序来检测这个函数，当 n=2，3，……，40 时，二个或多个人在同一天过生日的概率为多大？

5.21 用函数 random0 产生三个由随机数组的数组。三个数组分别包含 100，1000，2000 个元素。下一步，用函数 tic 和 tic 对三个数组用函数 ssort 进行排序计时。随着元素数目的增加，排序消耗的时间如何变化？

5.22 正态分布

由 random0 产生的随机变量符合平均分布。另一种分布类型是正态分布（如图 5.9 所示）。如果一个正态分布的平均数为 0，标准差为 1.0，那么这个正态分布被称为标准正态分布。标准正态分布的公式为

$$p(x)=\frac{1}{\sqrt{2\pi}}e^{-x^2/2} \qquad (5.14)$$



在 (−1, 1) 中遵守平均分布的随机变量能够产生遵守正态分布的随机变量。

1.在(-1,1)中任取遵守平均分布的随机变量 x_1 和 x_2，看 $tx_1^2+x_2^2<1$ 是否成立。如果成立用它们，如果不成立，则重试。

2.由下面公式得到的 y_1 和 y_2 将是正态分布随机变量。

$$y_1=\sqrt{\frac{-2\ln r}{r}}x_1 \qquad (5.15)$$

$$y_2=\sqrt{\frac{-2\ln r}{r}}x2 \qquad (5.16)$$

$$r=x_1^2+x_2^2 \qquad (5.17)$$

编写一个函数，每调用一次产生一个正态分布随机变量，通过得到 1000 个随机变量，计算他们的标准差，画出柱状图来检测你的函数。它们的标准差与 1.0 有多接近。

5.23 万有引力公式如下

$$F=G\frac{m_1m_2}{r^2} \qquad (5.18)$$

其中 G 是引力常量，大小为 $6.672\times10^{-11}\text{Nm}^2/\text{kg}^2$，$m_1$ 和 m_2 是两物体的质量，单位为 kg，r 为两质点的间距，单位为 m。编写一个程序，已知两物体的质量和距离，计算它们之间的万有引力。在距地表 38000 米的高空重 800kg 的卫星与地球之间的万有引力为多少？用这个数据来检测你的程序

5.24 瑞利分布

瑞利分布是另一种在许多现实问题中出现的随机变量分布类型。符合瑞利分布的随机变量可以由两个符合正态分布的随机变量通过计算得到。计算方法如下所示，n_1 和 n_2 是符合正态分布的随机变量。

$$r=\sqrt{n_1^2+n_2^2} \qquad (5.19)$$

A.创建一函数 rayleigh(n,m)，它将返回一个 n×m 的数组，数组元素符合瑞利分布。如果只有一个输入参数 n，它将会产生一个 n 阶方阵。确保你设计的函数能够检测输入参数，并为 **MATLAB** 帮助系统提供适当文本。

B.通过产生 20000 个符合瑞利分布的随机变量。并画出它们的分布的柱状图。这个分布看起来像什么？

C.计算出这些随机变量的平均数和标准差

5.25 恒虚警率(CFAR)

图 5.10a 显示的是一个简易的雷达接收器。当一个信号被接受器接受，此信号中将包括由目标返回的有用信息，还有一些热噪声。当信号被发现处理后，我们就能够从热噪声背景中挑拣出有用信号。我们可以设定一个阈值，如果信号超过了这个阈值，那么就声明发现了目标。不好的是，接受的噪声也会偶然的超过阈值。如果这种情况发生，噪声被误认为目标，

我们称之一个**虚警**。阈值要设得尽可能低，这样可以发现目标的微弱信号，但它又不能设得太低，这样我们就会得到许多的虚警。

在视频发现过后，这个接受机的热噪声符合瑞利分布。图 5.10b 显示的是平均振幅为 10V 的瑞利分布噪声的 100 个抽样，注意当发现阈值为 26 时，只有一次虚警。这些噪声抽样的概率分布如图 5.10c 所示。

发现阈值一般是噪声平均数的倍数，如果噪声水平改变，发现阈值也应随之改变，以控制虚警。这就是我们所说的恒虚警率发现。发现阈值单位一般有分贝。用 dB 表示的阈值和用电压表示的阈值关系如下：

$$\text{Threshold(volts)} = \text{Mean Nosise Level(volts)} \times 10^{\frac{dB}{20}} \tag{5.20}$$

或者是

$$\text{dB} = 20\log_{10}\left(\frac{\text{Threshold(volts)}}{\text{Mean Noise Level(volts)}}\right) \tag{5.21}$$

已知发现阈值，恒虚警率可由下面的公式求得。

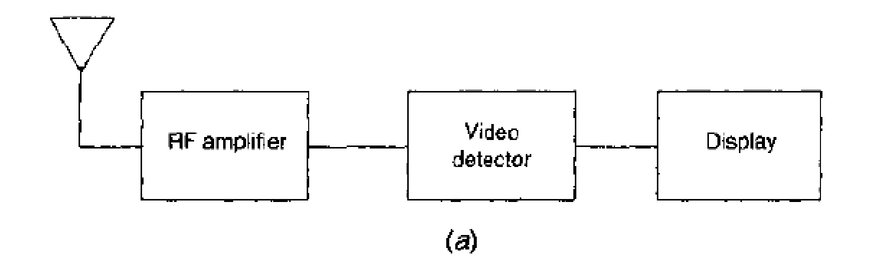
$$P_{fa} = \frac{\text{Number of False Alarms}}{\text{Total Number of Samples}} \tag{5.22}$$

编写一个程序，产生 1000000 个随机噪声抽样，这些抽样的平均幅度为 10V，并遵守瑞利分布。当发现阈值分别超出平均噪声水平时 5,6,7,8,9,10,11,12 和 13dB 时，它的恒虚警率为多少。当发现阈值被设为多少时，恒虚警率为 10^{-4}。

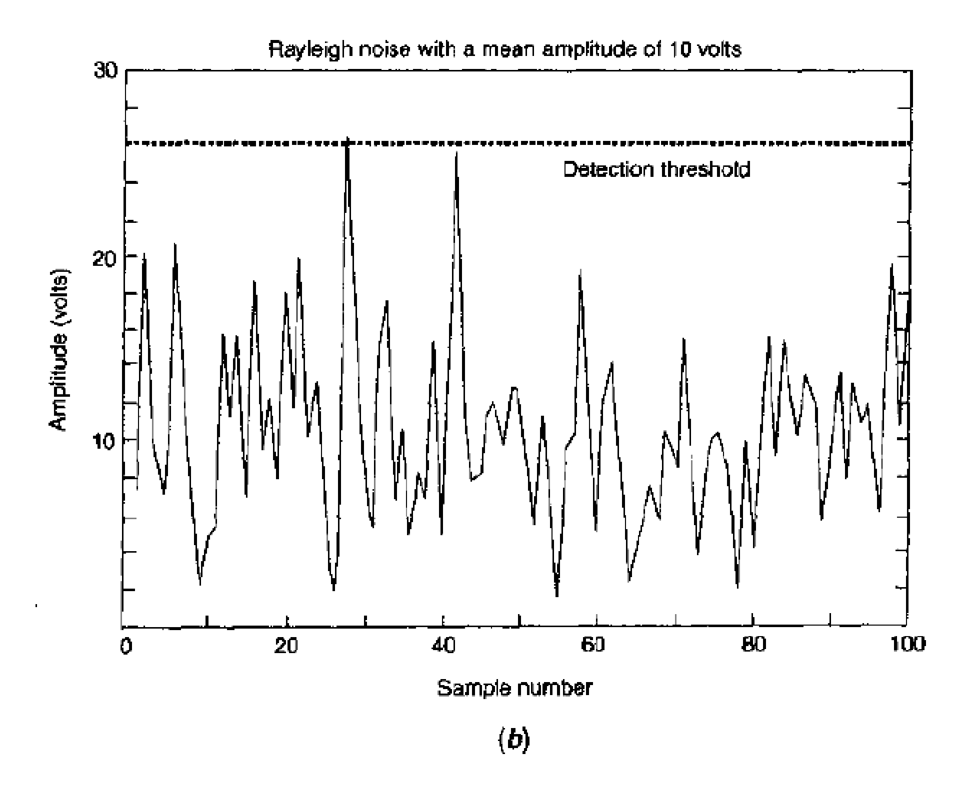
5.26 发现概率和虚警概率返回一个雷达目标的信号强度一般会在一定的时间内削弱。如要信号强度超过阈值，目标就会被发现。发现目标的概率可由以下公式计算。

$$P_d = \frac{\text{Number of Target Detections}}{\text{Total Number of Looks}} \tag{5.23}$$

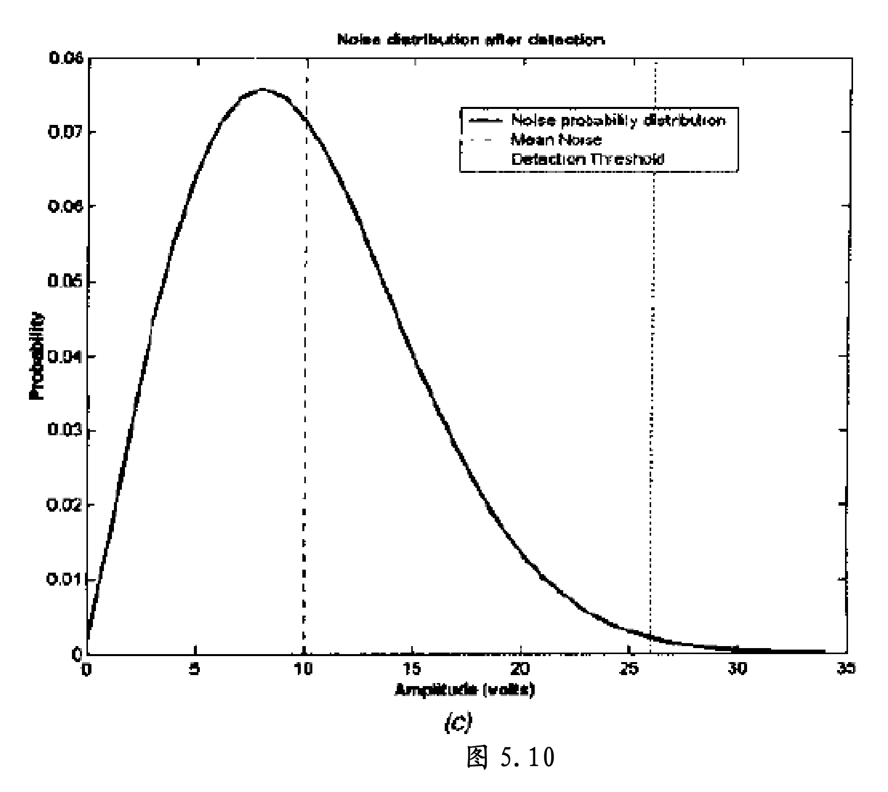
假设在特定位置的雷达不断地发射信号。在每一次发射信号时，在 10km 到 20km 的范围内被分为 100 个独立的范围抽样。这些抽样中的一个是目标，它的幅度符合正态分布，平均幅度为 7V，标准差为 1V。这 100 个抽样中包含有系统噪声，噪声幅度符合瑞利分布。平均幅度为 2V。当发现阈值分别为 8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 11.5 和 12.0dB 时，它的发现概率和虚警概率分别为多少？这个雷达的发现阈值应为多少？


RF amplifier
Video detector
Display

(a)


Rayleigh noise with a mean amplitude of 10 volts
Detection threshold
Amplitude (volts)
Sample number

(b)


Noise distribution after detection
Noise probability distribution
Mean Noise
Detection Threshold
Probability
Amplitude (volts)

(c)

图 5.10

第六章 复数数据、字符数据和附加画图类型

在第二章中，我们学习了 **MATLAB** 基础数据类型：double 和 char。**MATLAB** 还有许多的附加数据类型，在本章，我们将会了解它们中的一个。我们要讨论的附加数据类型是 **MATLAB** 支持的复数数据。我们也将学习如何使用 char 数据类型，以及如何把 **MATLAB** 数组扩展为多维数组。

本章还会涉及到 **MATLAB** 的附加画图类型。

6.1 复数数据

复数是指既包含实部又包含虚部的数。复数出现在许多的科研工作问题上。例如，在电器工程中，我们可以用复数代表交变电压，交变电流和阻抗。描述电器系统行为的公式经常用到复数。因为这是非常常见的，作为一个程师如果没有很好理解和运用复数，它无法工作。

复数的一般形式如下：

$$C=a+bi$$

其中 C 为复数，a 和 b 均为实数，i 代表$\sqrt{-1}$。a，b 分别为 C 的实部和虚部。由于复数有两个部分，所以它能在平面内标出。这个平面的横轴是实轴，纵轴是虚轴，所以复数在这个平面内为一个点，横轴为 a，纵轴为 b。用上面的方式表示一个复数，叫做直角坐标表示，为坐标的横轴与虚轴分别代表复数的实部与虚部。

复数有在一平面内另一种表达方式，既极坐标表示，公式如下，

$c = a + bi = z\angle\theta$

其中 z 代表向量的模，θ 代表辐角。直角坐标中的 a，b 和极坐标 z，θ 之间的关系为

$$a = z\cos\theta \qquad (6.2)$$

$$b = z\sin\theta \qquad (6.3)$$

$$z = \sqrt{a^2+b^2} \qquad (6.4)$$

$$\theta = \tan^{-1}\frac{b}{a} \qquad (6.5)$$

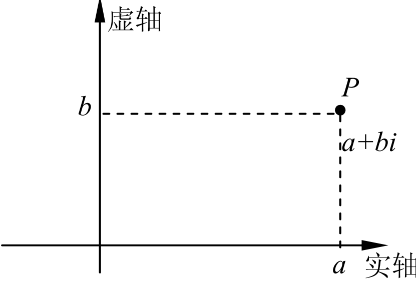


图 6.1 直角坐标系中复数

图 6.2 极坐标系中复数

MATLAB 用直角坐标表达复数。每一个复数应有一对实数（a，b）组成。第一个数（a）代表复数的实部，第二个数（b）代表复数的虚部。

如果复数 $c_1=a_1+b_1i$ 和复数 $c_2=a_2+b_2i$，那么它们的加减乘除运算定义如下。

$$c_1 + c_2 = (a_1 + a_2) + (b_1 + b_2)i \tag{6.6}$$

$$c_1 - c_2 = (a_1 - a_2) + (b_1 - b_2)i \tag{6.7}$$

$$c_1 \times c_2 = (a_1a_2 - b_1b_2) + (a_1b_2 + b_1a_2)i \tag{6.8}$$

$$\frac{c_1}{c_2} = \frac{(a_1a_2+b_1b_2)}{(a_1^2 + b_2^2)} + \frac{(b_1a_2-a_1b_2)}{(a_2^2+b_2^2)} i \tag{6.9}$$

当两个复数进行二元运算，**MATLAB** 将会用上面的法则进行加法，减法，乘法和除法运算。

6.1.1 复变量（complex variables）

当复数值赋值于一个变量名，**MATLAB** 将自动创建一个复变量。创建复数的最简单方法是用 **MATLAB** 本自带的因有变量 i 或 j，它们都被预定义为$\sqrt{-1}$ 。例如下面的语句将复数 4+3i 赋值于 c_1。

```
>> c1 = 4 + 3*i
c1 =
    4.0000 + 3.0000i
```

函数 isreal 可以判断一个数组包是实数组还是复数组。如果一个数组中的所有元素只有虚部，那么这个数组是复数组，并且 isreal(array)将会返回一个 0。

6.1.2 带有关系运算符的复数的应用

用关系运算符==来判断两复数是否相等，或用关系运算符~=判断两复数是否不相等，这种情况是可能的。这些运算都会产生出我们所期望的结果。例如，如果 c1=4+3i 和 c2=4-3i，那么关系运算 c1==c2 将会产生 0，关系运算 c1~=c2 将会产生 1。

但是，比较运算符>，<，<=或>=将不会产生我们所期望的结果。当复数进行此类关系运算时，只对复数的实部进行比较。例如，如果 c1=4+i3 和 c2=4+i8，那么比较运算 c1>c2 将会产生 1，尽管 c1 的模要比 c2 的模小。

如果我们需要用这些运算对两复数进行比较，我们更加关心的是两复数的模，而不只是实部。复数的模可以由 abs 固有函数计算得到（在下一节介绍，或者由公式（6.4）得到）。

$$|c| = \sqrt{a^2+b^2} \tag{6.4}$$

如果我们对两复数进行比较，得到的结果将更加合理。abs(c1)>abs(c2)将会产生 0，因为 c1 的模大于 c2 的模。

编程隐患

当我们应用关系运算符对复数运算时，一定要小心。关系运算符>，<，<=或>=只比较复数的实部，而不是它们的模。如果你要用这些关系运算符对一复数进行运算，比较两复数的模将更加常见。

6.1.3 复函数（complex function）

MATLAB 中有许多的函数支持复数的运算。这些函数可分为三大类。

1. 类型转换函数

这些函数把数据从复数据类型转换为实数数据类型（double）。函数 real 将复数的实部转化为 double 型数据，把复数的虚部抛弃。函数 imag 把函数的虚部转化为相应的实数。

2. 绝对值和幅角函数

这些函数把复数转化它的极坐标形式。函数 abs(c)用于计算复数 c 相应的绝对值，公式如下

$$abs(c)=\sqrt{a^2+b^2}$$

其中 c=a+bi。函数 angle(c)用下面的公式计算复数 c 的幅角

angle(c) = atan2(imag(c), real(c))

由它产生的角的取值范围为-π<θ≤π。

表 6.1 常见的支持复数运算的 **MATLAB** 函数

函数	描述
conj(c)	计算 c 的共共轭复数。如果 c=a+bi，那么 conj(c)=a-bi。
real(c)	返回复数 c 的实部
imag(c)	返回复数 c 的虚部
isreal(c)	如果数组 c 中没有一个元素有虚部，函数 isreal(c)将返回 1。所以如果一个数组 c 是复数组成，那么~isreal(c)将返回 1。
abs(c)	返回复数 c 模
angle(c)	返回复数 c 的幅角，等价于 atan2(imag(c)，real(c))

3. 数学函数

许多的数函数都可以对复数进行运算。这些函数包括指数函数，对数函数，三角函数，还有平方根函数。函数 sin，cos，log，sqrt 等既能对复数数据进行运算，又能对实数据进行运算。

一些支持复数运算的函数在表 6.1 中列出。

例 6.1

二次方程的求解（重写）

复数的价值体现在它能使运算简化。

例如，我们在例 3.2 中已解决的二次方程的求解问题，但它根据判别式用到 3 个选项的选择结构，由于复数的出现，负数的平方根的处理将不困难。所以能够大大简化我们的计算。编写一个普通的程序，解一元二次方程的根，不管是什么类型的。用复变量，而不用选择结构。

1. 陈述问题

编写一个程序，解一元二次方程的根，不管有两个不同的实根，还是用两个相同的实根或两个不同复根。不需要检测判别式。

2. 定义输入输出

本程序所需要方程式

$ax^2 + bx + c = 0$ (3.1)

的三个系数 a，b，c。输出是这个方程式的所有根。

3. 设计算法

这个程序从整体上可以分为三大步，即输入，计算，输出

```
Read the input data
Calculate the roots
Write out the roots
```

我们现在把每一步进行逐步细化。这时判别式的值对程序的执行过程不产生影响。伪代码如下：

```
Prompt the user for the coefficients a, b, and c.
Read a, b, and c
discriminant ← b^2 - 4 * a * c
x1 ←( -b + sqrt(discriminant) ) / (2*a)
x2 ←( -b - sqrt(discriminant) ) / (2*a)
Print 'x1 = ' , real(x1), ' + i ', imag(x1)
Print 'x2 = ' , real(x2), ' + i ', imag(x2)
```

4. 将算法转化为 **MATLAB** 语句

```
% Script file: calc_roots2.m
%
% Purpose:
% This program solves for the roots of a quadratic equation
% of the form a*x**2 + b*x + c = 0. It calculates the answers
% regardless of the type of roots that the equation possesses.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/06/98 S. J. Chapman Original code
%
% Define variables:
% a -- Coefficient of x^2 term of equation
% b -- Coefficient of x term of equation
% c -- Constant term of equation
% discriminant -- Discriminant of the equation
% x1 -- First solution of equation
% x2 -- Second solution of equation
% Prompt the user for the coefficients of the equation
disp ('This program solves for the roots of a quadratic ');
disp ('equation of the form A*X^2 + B*X + C = 0. ');
a = input ('Enter the coefficient A: ');
b = input ('Enter the coefficient B: ');
c = input ('Enter the coefficient C: ');
% Calculate discriminant
discriminant = b^2 - 4 * a * c;
% Solve for the roots
x1 = ( -b + sqrt(discriminant) ) / ( 2 * a );
x2 = ( -b - sqrt(discriminant) ) / ( 2 * a );
% Display results
disp ('The roots of this equation are:');
fprintf ('x1 = (%f) +i (%f)\n', real(x1), imag(x1));
fprintf ('x2 = (%f) +i (%f)\n', real(x2), imag(x2));
```

5.检测程序下一步，我们必须输入检测来检测程序。我们要有三组数据进行检测，其判别式分别大于 0，等于 0，小于 0。根据方程式（3。1），用下面的方程式验证程序。

$x^2 + 5x + 6 = 0$　　　　$x = -2, x = -3$
$x^2 + 4x + 4 = 0$　　　　$x = -2$
$x^2 + 2x + 5 = 0$　　　　$x = -1 \pm 2i$

我们把它们的系数分别输入程序，结果如下

```
>> calc_root2
This program solves for the roots of a quadratic
equation of the form A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 5
Enter the coefficient C: 6
The roots of this equation are:
x1 = (-2.000000) +i (0.000000)
x2 = (-3.000000) +i (0.000000)
>> calc_root2
This program solves for the roots of a quadratic
equation of the form A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 4
Enter the coefficient C: 4
The roots of this equation are:
x1 = (-2.000000) +i (0.000000)
x2 = (-2.000000) +i (0.000000)
>> calc_root2
This program solves for the roots of a quadratic
equation of the form A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 2
Enter the coefficient C: 5
The roots of this equation are:
x1 = (-1.000000) +i (2.000000)
x2 = (-1.000000) +i (-2.000000)
```

在三种不同的情况下，程序均给出了正确的结果。注意此程序与例 3.2 中的程序相比有多简单。复数数据的应用可大大简化我们的程序。

6.1.4 复数数据的作图

因为复数数据既包括实部又包括虚部，所以在 **MATLAB** 中复数数据的作图与普通实数据的作图有所区别。例如，考虑下面的函数

$$y(t) = e^{-0.2t}(\cos t + i \sin t) \tag{6.10}$$

如果我们用传统的 plot 命令给这个函数作图，只有实数数据被作出来，而虚部将会被忽略。下面的语句得到图象如图 6.3 所示，注意出现了警告信息：数据的虚部被忽略

```
t = 0:pi/20:4*pi;
y = exp(-0.2*t) .* (cos(t) + i * sin(t));
plot(t, y);
title('\bfPlot of Complex Function vs Time');
xlabel('\bf\itt');
ylabel('\bf\ity(t)');
```

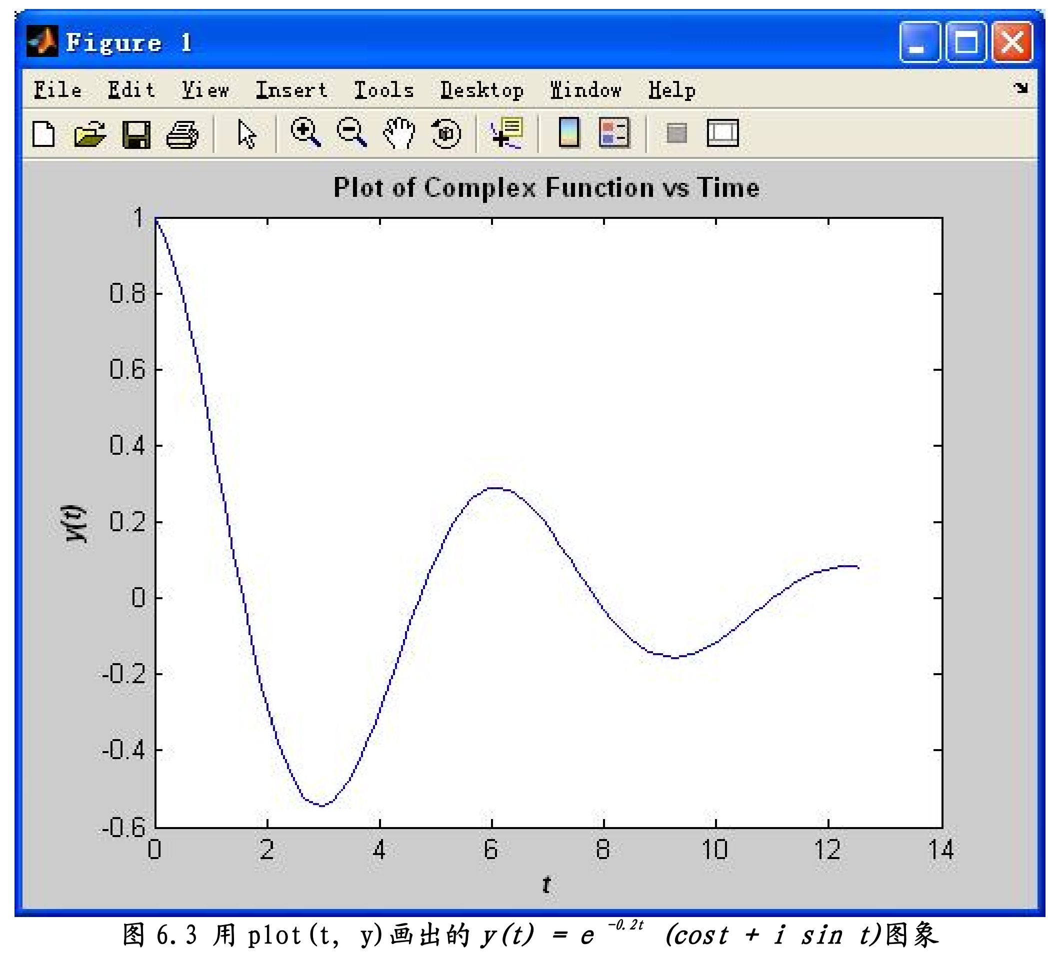


图 6.3 用 plot(t, y)画出的 $y(t) = e^{-0.2t}(\cos t + i \sin t)$图象

如果函数的实部和虚部都需要的话，那么用户可以有几种选择。我们可以用下面的语句，在相同的时间轴内画出函数的图象（图 6.4）。

```
t = 0:pi/20:4*pi;
y = exp(-0.2*t) .* (cos(t) + i * sin(t));
plot(t, real(y),'b-');
hold on;
plot(t, imag(y),'r--');
title('\bfPlot of Complex Function vs Time');
xlabel('\bf\itt');
ylabel('\bf\ity(t)');
legend('real','imaginary');
hold off;
```

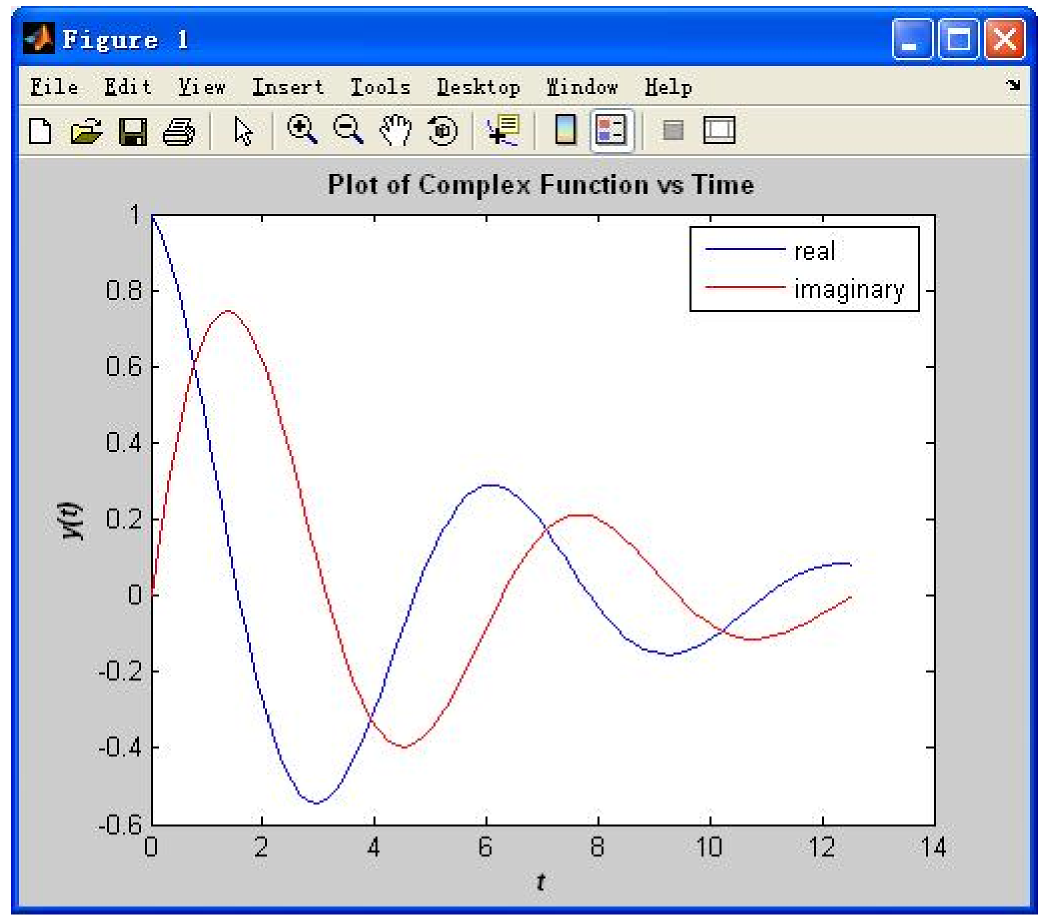


图 6.4 包含了 y(t)的实部和虚部

可选择的，函数的实部-虚部图可以被画出来。如果有一个复参数提供给 plot 函数它会自动产生一个函数的实部-虚部图。产生这类图的语句如下，产生的结果如图 6.5 所示。

```
t = 0:pi/20:4*pi;
y = exp(-0.2*t) .* (cos(t) + i * sin(t));
plot(y,'b-');
title('\bfPlot of Complex Function');
xlabel('\bfReal Part');
ylabel('\bfImaginary Part');
```

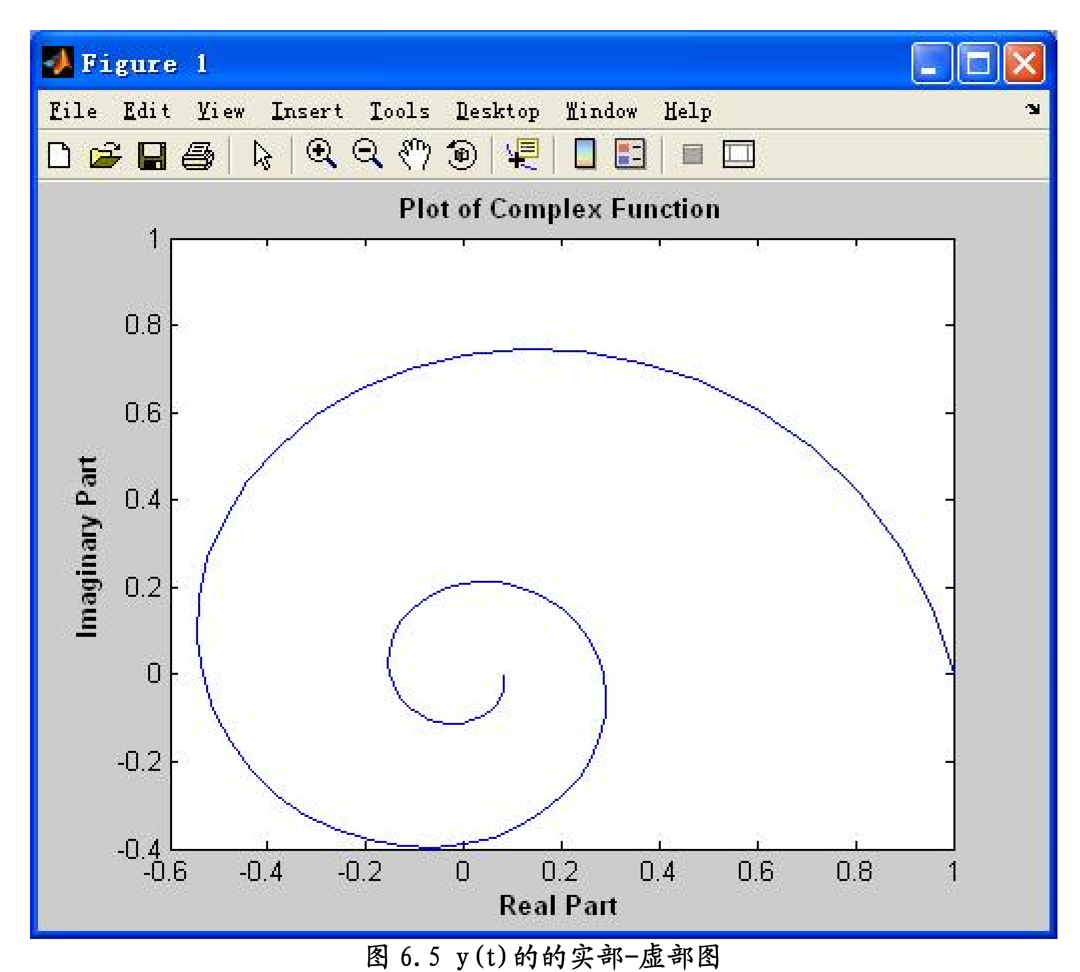


图 6.5 y(t)的的实部-虚部图

最后，我们可以画出函数的极坐标图。产生这类图语句如下，产生的结果如图图 6.6 所示。

```
t = 0:pi/20:4*pi;
y = exp(-0.2*t) .* (cos(t) + i * sin(t));
polar(angle(y),abs(y));
title('\bfPlot of Complex Function');
```

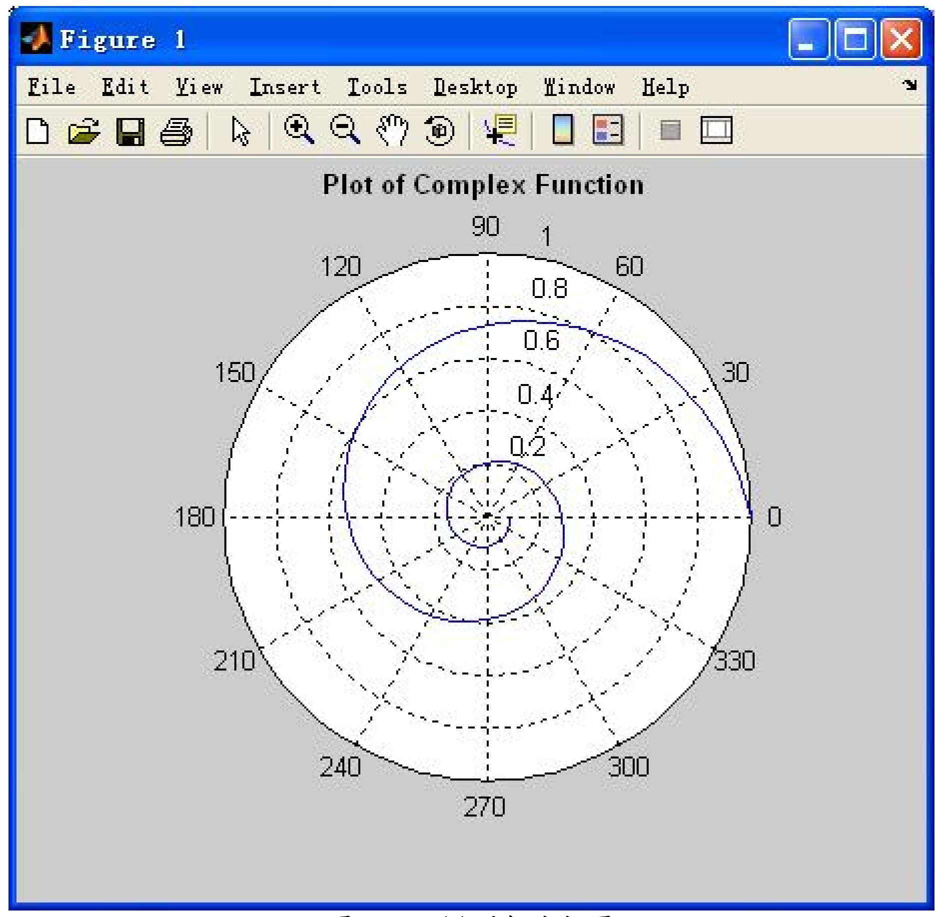


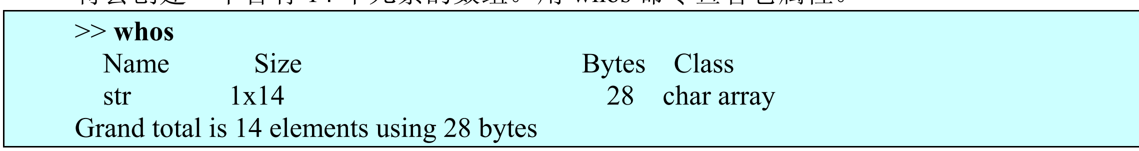
图 6.6 y(t)的极坐标图

6.2 字符串函数（string functions）

一个 **MATLAB** 字符串是一个 char 型数组。每一个字型占两个字节。当字符串被赋值于一个变量时，这个变量将被自动创建为字符变量。例如语句

```
str = 'This is a test';
```

将会创建一个含有 14 个元素的数组。用 whos 命令查看它属性。



```
>> whos
  Name          Size                    Bytes  Class
  str           1x14                       28  char array
Grand total is 14 elements using 28 bytes
```

一个专门的函数 ischar 常用来判断一个变量是否为字符数组。如果是的话，那么函数较会返回 1，如果不是，将会返回 0。

在下面的的小节中，我们将向大家介绍一些对字符串进行操作的函数。

6.2.1 字符转换函数

我们可以利用 double 函数把变量从字型转化为 double 型。所以，函数 double(str)产生的结果为

```
>> x = double(str)
```

```
x =
  Columns 1 through 12
    84   104   105   115    32   105   115    32    97    32   116   101
  Columns 13 through 14
   115   116
```

我们可以利用 char 函数把 double 型数据转化为字符型数据。所以函数 char(x)产生的结果为

```
>> x = char(x)
x =
This is a test
```

6.2.2 创建二维字符数组

我们可以创建二维字符数组，但一个数组中每一行的长度都必须相等。如果其中的一行比其他行短，那么这个字符数据将会无效，并产生一个错误。例如，下面的语句是非法的，因为他两行的长度不同。

```
name = ['Stephen J. Chapman'; 'Senior Engineer'];
```

创建二维字符数组的最简单的方法是用 char 函数。函数将会自动地寻找所有字符串中最长的那一个。

```
>> name = char('Stephen J. Chapman','Senior Engineer')
name =
Stephen J. Chapman
Senior Engineer
```

二维字符数组也可以用函数 strvcat，这个函数我会在下一节中介绍。

好的编程习惯
用 char 函数创建二维字符数组，我们就不用担心每一行的长度不相同了。

我们可以应用 deblank 函数去除多余空格。例如，下面的语句去除 name 数组中第二行的多余空格，产生的结果与原来的进行比较。

```
>> line2 = name(2,:)
line2 =
Senior Engineer
>> line2_trim = deblank(name(2,:))
line2_trim =
Senior Engineer
>> size(line2)
ans =
     1    18
>> size(line2_trim)
ans =
     1    15
```

6.2.3 字符串的连接

函数 strcat 水平连接两字符串，忽略所有字符串末端的空格，而字符串的空格保留。例如，下面的语句为

```
>> result = strcat('string 1   ','String 2')
```

```
result =
string 1String 2
```

产生的结果 string 1String 2。

函数 strvcat 用于竖直地连接两字符串，自动地把它转化为二维数组。这个函数将产生这样的结果

```
>> result = strvcat('Long String 1    ','String 2')
result =
Long String 1
String 2
```

6.2.4 字符串的比较

字符串与子字符串可以通过下面许多的方式进行比较。

- 两个字符串，或两个字符串的部分，看两者是否相同
- 两个独立的字符相比较看两者是否相同
- 检查字符串判断每一个字符是字母，还是空格

6.2.4.1 比较两字符串，看是否相同

你可以利用 **MATLAB** 函数比较两字符串整体是否相同。它们是

- strcmp　　判断两字符串是否等价
- strcmpi　　忽略大小写判断两字符串是否等价
- strncmp　　判断两字符串前 n 个字符是否等价
- strncmpi　　忽略大小写判断两字符串前 n 个字符是否等价

函数 strcmp 比较字符串，包括字符串前面或后面的空格。如果两字符串完全相同，那么这个函数将返回 1。否则，返回 0。strcmpi 与 strcmp 类似，但它忽略了大小写（即“a”与“A”看作相同的）

函数 strncmp 用来比较两字符串前 n 个字符串，包含开头的空格，如果这个 n 个字符是相同的，它们将会返回 1。否则它将会返回 0。函数 strncmpi 与它相类似，但忽略了大小写。

为了更好的理解这些函数，考虑下面的字符串

```
str1 = 'hello';
str2 = 'Hello';
str3 = 'help';
```

字符串 str1 和 str2 不相同，但它第一个字母大小不同。所以 strcmp 将返回 0，strcmpi 将返回 1。

```
>> c = strcmp(str1,str2)
c =
     0
>> c = strcmpi(str1,str2)
c =
     1
```

字符串 str1 和 str3 不相同，所以 strcmp 与 strcmpi 返回 0。但是 str1 和 str3 的前三个字符是相同，所以按照下面的方式调用将会返回 1。

```
>> c = strncmp(str1, str3, 2)
c =
     1
```

6.2.4.2 判断单个字符是否相等

我们可以利用**MATLAB**关系运算符对字符数组中的每一个元素进行检测，看是否相同，但是我们要保证它们的维数是相同的，或其中一个是标量。例如，你可以用相等运算符（==）来检测两字符串是否相匹配。

```
>> a = 'fate';
>> b = 'cake';
>> result = a == b
result =
     0     1     0     1
```

所有的关系运算符（>，>=，<，<=，==，~=）都是对字符所对应的 ASCII 值进行比较。

与 C 语言不同，**MATLAB** 中没有一个内建函数，对两字符串在整体进行“大于”或“小于”的关系运算。我们将会本节末创建一个类似的函数。

6.2.4.3 在一字符串内对字符进行判断

有两个函数可对一个字符串内的字符逐个进行分类。

- isletter　　用来判断一个字符是否为字母
- isspace　　判断一个字符是否为空白字符（空格，tab，换行符）

例如，我们要创建一个字符串 mystring，

```
mystring = 'Room 23a'
```

函数 isletter 检测字符串中的每一个字符，将产生一个与字符串 isletter 相同长度输出向量，一个字符对应一个 1。

```
>> a = isletter(mystring)
a =
     1     1     1     1     0     0     0     1
```

在 a 中前四个元素和最后一个元素是 1，因为它们对应的 mystring 中的字符是字母。函数 isspace 检测字符串中的每一个字符，将产生一个和字符串长度相同的输出变量，对应于空字符的向量元素为 0。

因为向量的第五个元素对应的是空格，所以向量的第五个元素的值为 1。

6.2.5 在一个字符串中查找/替换字符

MATLAB 提供了许多的函数，用来对字符串中的字符进行查找或替换。考虑字符串 test

test = 'This is a test!';

函数 findstr 返回短字符串在长字符串中所有的开始位置。例如为了寻找 test 内的所有“is”

```
>> position = findstr(test,'is')
position =
     3     6
```

字符串“is”在 test 内出现两次，开始位置分别为 3 和 6。

函数 strmatch 是另一种匹配函数。它用来查看二维数组行开头的字符，并返回那些以指定的字符序列为开头行号。它的基本形式如下

```
result = strmatch(str,array);
```

例如，我们用 strvcat 创建一个二维数组，

```
array = strvcat('maxarray','min value','max value');
```

那么下面的语句将会返回开始字符为“max”的行数。

```
>> result = strmatch('max',array)
result =
     1
     3
```

函数 strrep 用于进行标准的查找和替换操作。它能找到一个字符串中的所有另一个字符串，并被第三个字符串替换。这个函数形式为

```
result = strrep(str,srch,repl)
```

其中 str 是被检测的字符串，srch 是要查找到的字符串，repl 是用于替代的字符串，例如，

```
>> result = strrep(test,'test','pest')
result =
This is a pest!
```

函数 strtok 返回输入字符串中第一次出现在分隔符前面的所有字符。默认的分隔符为一系列的空白字符。strtok 的形式如下

[token, remainder] = strtok(string,delim)

其中 string 是输入字符串，delim 是可选择的分隔符，token 代表输入字符串中第一次出现在分隔符前面的所有字符，remainder 代表这一行的其余部分。例如

```
>> [token, remainder] = strtok('This is a test!')
token =
This
remainder =
 is a test!
```

你可以利用函数 strtok 把一个句子转换为单词。例如，下面的代码从字符数组 input_string 中分离出每一个单词，并把每一个单词独立地存储在字符数组 all_words 的每一行中。

```
function all_words = word(input_string)
remainder = input_string
all_words = '';
while (any(remainder))
    [chopped, remainder] = strtok(remainder);
    all_words = strvcat(all_words, chopped);
end
```

6.2.6 大小写转换

函数 upper 和 lower 分别把一个字符串中所有转化大定和小写。例如

```
>> result = upper('This is test 1!')
result =
THIS IS TEST 1!
>> result = lower('This is test 2!')
result =
this is test 2!
```

注意在大小转换时，数字和符号不受影响。

6.2.7 字符串转换为数字

MATLAB 把由数字组成的字符串转化为数字要用到函数 eval。例如，字符串“3.141592”能用下面的语句把它转换为数字。

```
>> a = '3.141592';
>> b = eval(a)
b =
      3.1416
>> whos
   Name         Size                    Bytes   Class
   a            1x8                        16   char array
Grand total is 8 elements using 16 bytes
```

字符串可以用 sscanf 函数转化为数字。这个函数根据格式化转义字符转化为相应的数字。这个函数最简单的形式如下

```
value = sscanf(string, format)
```

其中，string 是要转化的字符串，format 是相应的转义字符。函数 sscanf 两种最普通的转义序是“%d”，“%g”，它们分别代表输出为整数或浮点数。这个函数更多的细节我们将在第 8 章介绍。在作图中，创建一个复杂的标题或标签，它是非常有用的。

下面的例子用于说明函数 sscanf 的应用。

```
>> value1 = sscanf('3.141593','%g')
value1 =
      3.1416
>> value2 = sscanf('3.141593','%d')
value2 =
        3
```

6.2.8 数字转化为字符串

MATLAB 中有许多的字符串/数字转换函数把数字转化为相应的字符串。我们在这里只看两个函数 num2str 和 int2str。考虑标量 x

```
x = 5317;
```

在默认的情况下，**MATLAB** 把数 x 作为一个 1×1 的 double 数组，它的值为 5317。函数 int2str（integer to string）所这个标量转化为 1×4 的字符数组，包含有字符串“5317”。

```
>> x = 5317;
>> y = int2str(x);
>> whos
   Name         Size                    Bytes   Class
   x            1x1                         8   double array
   y            1x4                         8   char array
Grand total is 5 elements using 16 bytes
```

函数 num2str 为输出字符串的格式提供更多的控制。第二个可选择的参数可以对输出字符串的数字个数进行设置或指定一个实际格式。例如

```
>> p = num2str(pi,7)
p =
3.141593
>> p = num2str(pi,'%10.5e')
```

```
p =
3.14159e+000
```

函数 int2str 和 num2str 对作图标签是非常有用的。例如，下面的语句用 num2str 生成图象的标签。

```
function plotlabel(x,y)
plot(x,y)
str1 = num2str(min(x));
str2 = num2str(max(x));
out = ['Value of f from ' str1 ' to ' str2];
xlabel(out);
```

还有一些转换函数，用于把数字值从十进制转化另一种数制，例如二进制或十六进制。例如函数 dec2hex 把一个十进制数转化为相应的十六进制字符串。此类的函数还有 hex2num，hex2dec，bin2dec，dec2bin，base2dec，你可以通过 **MATLAB** 网上帮助来获取这些函数的作用和使用方法。

MATLAB 函数 mat2str 可以把一个数组转化为相应的 **MATLAB** 能运算字符串。这个字符串可以是 eval 函数的输入，函数 eval 对这个字符串的运算和直接在命令窗口键入效果是一样的。例如，我们定义一个数组如下

```
>> a = [1 2 3; 4 5 6]
a =
     1     2     3
     4     5     6
```

函数 mat2str 运行得到的结果为

```
>> b = mat2str(a)
b =
[1 2 3;4 5 6]
```

最后，**MATLAB** 中有一个专门的函数 sprintf 等价于函数 fprintf，唯一不同的是它的输出是一个字符串。这个函数对字符串的格式化操作的完全支持。例如，

```
>> str = sprintf('The value of pi = %8.6f',pi)
str =
The value of pi = 3.141593
```

在图象中，用这些函数创建复杂的标题或标签将会非常的有用。

6.2.9 总结

普通的 **MATLAB** 字符串函数总结在表 6.2 中。

表 6.2 普通的 **MATLAB** 字符串函数

类别	函数	描述
普通		
	char	(1)把数字转化为相应的字符值
		(2)把二维数组转化相应的字符串
	double	把字符转化为相应的 double 值
	blanks	创建一个由空格组成的字符串
	deblanks	去除字符串末端的空格
字符检测		

ischar	如果是一个字符数组，那么将会返回 1
isletter	如果是字母表中的字母，那么将会返回 1
isspace	如果是空白字符，那么将会返回 1
字符串操作	
strcat	连接字符串
strvcat	竖直地连接字符串
strcmp	如果两字符串相等，那么函数将会返回 1
stricmp	忽略大小写如果两字符串相等，那么函数将会返回 1
strncmp	如果两字符串的前 n 个字母相等，那么函数将会返回 1
strncmpi	忽略大小，如果两字符串的前 n 个字母相同，那么数将会返回 1
findstr	在一个字符串中寻找另一个字符串
strfind	在一个字符串中寻找另一个字符串（版本 6.1 或以后的版本）
strjust	对齐字符串
strmatch	找字符串的区配
strrep	用一个字符串去替代另一个字符串
strtok	查找一字符串
upper	把字符串的所有字符转化为大写
lower	把字符串的所有字符转化为小写
数字转化为字符串	
int2str	把整数转化为相应的字符串形式
num2str	把数字转化为相应的字符串形式
mat2str	把矩阵转化为相应的字符串形式
sprintf	对一字符串进行格式化输出
字符串转化为数字	
str2double	把字符串转化相应的 double 型数据
str2num	把字符转化成数字
sscanf	从字符串中读取格式化数据
数制转换	
hex2num	把 IEEE 十六进制字符型型数据转化为 double 形数据
hex2dec	把十六制字符串转化为相应的十进制整数
dec2hex	把十进制数转化为相应的十六制字符串
bin2dec	把二进制字符串转化为相应的十进制整数
base2dec	把 base B 转化为相应的十进制数据
dec2base	把十进制转化为相应的 base B
hex2num	把 IEEE 十六进制字符型型数据转化为 double 形数据

例 6.2

字符串比较函数

在 C 语言中，函数 strcmp 根据 Ascii 码中字符顺序（我们称之为字典顺序）比较两字符，如果第一个字符串的字典顺序在第二个字符串字典之后，函数将会产生-1，如果两字符串相同那么将会产生 0，如果第一个字符串的字典顺序在第二个字符串字典之前那么函数将会返回+1。

创建一个 **MATLAB** 函数 c_strcmp 用来比较两字符串，其功能与 C 语言 strcmp 中的函数功能相类似。这个函数对字符串进行比较时，应忽略末尾的空格。注意这个函数必须控制两字符串不同长的情况。

答案：

1．陈述问题

编写一个函数，比较两字符串 str1 和 str2，并返回以下结果

- -1　　如果 str1 在字典顺序比 str2 的晚
- 0　　如果两字符串的字典顺序相同
- +1　　如果 str1 的字典顺序比 str2 的早

2．定义输入输出量

函数所需的输入量为两个字符串，str1 和 str2。这个函数的输出为-1、0 或 1。

3．描述算法这个工程可分为以下四大步

```
Verify input strings
Pad strings to be equal length
Compare characters from beginning to end, looking for the first difference
Return a value based on the first difference
```

我们将以上每一大步分解成更小的更细的小块。第一，我们必须验证传递给函数的数据是正确的。函数必须有两个参数，且这两个参数必须为字符。这一步的伪代码为

```
% Check for a legal number of input arguments.
msg = nargchk(2, 2, nargin)
error(msg)
% Check to see if the arguments are strings
if either argument is not a string
    error('str1 and str2 must both be strings')
else
    (add code here)
end
```

下一步，我们要做的是把这两个字符串具有相同的长度。最简单的方法是用 strvcat 函数把这两个字符串联合成一个二维数组。注意在这个步骤中产生字符串末端空格只是为了两字符串相等，所以这些空格可以被省略。

```
% Pad strings
strings = strvcat(str1, str2)
```

现在我们要对字符串中的每一个字符进行比较，直到我们一个不同的字符出现，并基于这种不同返回相应的值。为了达到这个止目的，其中的方法是应用关系运算符比较两个字符串，产生一个由 0 和 1 组成的数组。然后我们可以寻找第一个 1，因为它两字符串在这里出现第一次不同。这一步的伪代码如下

```
% Compare strings
diff = strings(1, :) ~= strings(2, :)
if sum(diff) == 0
    % Strings match
    result = 0
else
    % Find first difference
    ival = find(diff)
    if strings(1, ival(1)) > strings(2, ival(1))
        result = 1
    else
        result = -1
    end
end
```

4．把算法转化为相应的 **MATLAB** 语句

```
function result = c_strcmp(str1,str2)
%C_STRCMP Compare strings like C function "strcmp"
% Function C_STRCMP compares two strings, and returns
% a -1 of str1 < str2, a 0 if str1 == str2, and a
% +1 if str1 > str2.
```

```
% Define variables:
% diff          -- Logical array of string differences
% msg           -- Error message
% result        -- Result of function
% str1          -- First string to compare
% str2          -- Second string to compare
% strings       -- Padded array of strings
% Record of revisions:
% Date      Programmer      Description of change
% ====      ==========      =====================
% 10/18/98  S. J. Chapman   Original code
% Check for a legal number of input arguments.
msg = nargchk(2,2,nargin);
error(msg);
% Check to see if the arguments are strings
if ~(isstr(str1) & isstr(str2))
    error('Both str1 and str2 must both be strings!')
else
    % Pad strings
    strings = strvcat(str1,str2);
    % Compare strings
    diff = strings(1,:) ~= strings(2,:);
    if sum(diff) == 0
        % Strings match, so return a zero!
        result = 0;
    else
        % Find first difference between strings
        ival = find(diff);
        if strings(1,ival(1)) > strings(2,ival(1))
            result = 1;
        else
            result = -1;
        end
    end
end
```

5. 检测程序

我们必须用多个字符串对程序进行检测

```
>> result = c_strcmp('String 1','String 1')
result =
     0
>> result = c_strcmp('String 1','String 1      ')
result =
     0
>> result = c_strcmp('String 1','String 2')
result =
    -1
>> result = c_strcmp('String 1','String 0')
result =
     1
>> result = c_strcmp('String 1','str')
result =
    -1
```

第一次检测返回 0，是因为两字符串是相同的。第二次也返回 0，因为两字符串也是相等的，只是末端空格不同，末端空格被忽略。第三次检测返回-1，因为两字符串第一次的不同出现在第 8 位上，且在这个位置上“1”<“2”。第四次检测将返回 1，因为两字符串第一

次的不同出现在第 8 位上，且在这个位置上，“1” > “0”。第五次检测将会返回-1，因为两字符串的第一个字符就不同，在 ascii 的序列上“S” < “s”。这个函数工作正常。

测试 6.1

本测试提供了一个快速的检查方式，看你是否掌握了 6.1 到 6.2 的基本内容。如果你对本测试有疑问，你可以重读 6.1 到 6.2，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. 下面的语句产生的 result 的值是多少？

```
(a) x = 12 + i*5;
    y = 5 - i*13;
    result = x > y;
(b) x = 12 + i*5;
    y = 5 - i*13;
    result = abs(x) > abs(y);
(c) x = 12 + i*5;
    y = 5 - i*13;
    result = real(x) - imag(y);
```

2. 果 array 是一个复数组，那么函数 plot(array)将会产生怎样的结果。

3. 们如何将一个 char 数据类型的向量转化为相应的 double 型数据类型的数据向量。从式 4 到 11，判断这些语句是否正确。如果它们正确，那么将产生什么结果？

```
4.  str1 = 'This is a test! ';
    str2 = 'This line, too.';
    res = strcat(str1, str2);
5.  str1 = 'Line 1';
    str2 = 'Line 2';
    res = strcati(str1,str2);
6.  str1 = 'This is a test!';
    str2 = 'This line, too.';
    res = [str1; str2];
7.  str1 = 'This is a test!';
    str2 = 'This line, too.';
    res = strvcat(str1, str2);
8.  str1 = 'This is a test!';
    str2 = 'This line, too.';
    res = strncmp(str1, str2, 5);
9.  str1 = 'This is a test!';
    res = findstr(str1, 's');
10. str1 = 'This is a test!';
    str1(4:7) = upper(str1(4:7));
11. str1 = 'This way to the egress.';
    str2 = 'This way to the egret.';
    res = strncmp(str1, str2);
```

6.3 多维数组

从，**MATLAB**5.0 版本开始支持多于二维的数组。这些多维数组用来显示多于二维的数据，或显示多个版本的二维图。例如，在一个三维空间中压力和速度的测量对于一些学科来说是非常重要的，例如空气动力学，流体力学。在这些领域中自然会用到多维数组。

多维数组是二维数组的扩展。每增加一维，它们所对应的每个元素就会多一个下角标。

我们可以轻易地创建一个多维数组。它既可以通过直接的赋值语句进行赋值，可用相同的函数进行创建（和一维二维中一样）。例如，假设你已经利用赋值语句创建了一个二维数组

```
>> a = [1 2 3 4; 5 6 7 8]
a =
     1     2     3     4
     5     6     7     8
```

这是一个 2×4 数组，每个元素被访问时，都应带有两个下标。这个数组可扩展为一个三维 2×4×3 数组，语句如下

```
>> a(:,:,2) = [9 10 11 12; 13 14 15 16];
>> a(:,:,3) = [17 18 19 20; 21 22 23 24]
a(:,:,1) =
     1     2     3     4
     5     6     7     8
a(:,:,2) =
     9    10    11    12
    13    14    15    16
a(:,:,3) =
    17    18    19    20
    21    22    23    24
```

在这个多维数组中的每一个元素都可以用它的函数名加上它的三个下标进行访问，数据下标的创建可以用克隆运算符。例如，a(2,2,2)的值为

```
>> a(2,2,2)
ans =
    14
```

向量 a(1,1,:)为

```
>> a(1,1,:)
ans(:,:,1) =
     1
ans(:,:,2) =
     9
ans(:,:,3) =
    17
```

多维数组也可以用与其他数据相同的函数进行创建

```
>> b = ones(4,4,2)
b(:,:,1) =
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
b(:,:,2) =
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
>> c = randn(2,2,3)
c(:,:,1) =
   -0.4326    0.1253
   -1.6656    0.2877
c(:,:,2) =
   -1.1465    1.1892
    1.1909   -0.0376
c(:,:,3) =
    0.3273   -0.1867
    0.1746    0.7258
```

多维数组的维数可以利用 ndims 函数得到，数组的大小可通过 size 函数得到。

```
>> ndims(c)
```

```
ans =
        3
>> size(c)
ans =
        2        2        3
```

如果你需要多数组编写应用程序，你可以通过阅读 **MATLAB** user's guide 来了解更多的多维数组函数的细节。

好的编程习惯
我们可以利用多维数组来解决自然界的多变量问题，如空气动力学和流体力学。

6.4 关于二维作图的补充说明

在前面的章节中，我们学习了如何创建线性图，对数图，线性-对数图和极坐标图。**MATLAB** 提供了许多的画图类型，用来显示你的数据。本节将向你介绍它们其中的一些操作。

6.4.1 二维作图的附加类型

除了我们已经看到图象类型，**MATLAB** 还支持其他的图象。实际上，在 **MATLAB** 帮助工作台中列出超过 20 种类型的作图。例如针头图（Stem Plots），阶梯图（stair plots），条形图，饼图（pie plots），罗盘图（compass plots）。在针头图中的每一个值都用一个圆圈和垂直于 x 轴的直线连接而成。在阶梯图中的每一个值都是用连续的竖直的长条线来表示，形成阶梯状效果。条形图可分成水平条形图和竖直条形图。

饼图用不同的扇区代表不同的变量。最后罗盘图是另一种极坐标图它的每一值用箭头来表示。

针头图，阶梯图，条形图，饼图，罗盘图与普通的图象差不多，它的调用方式相同。例如，下面显示的是一个针头图的代码，产生的图象如图 6.7a 所示。

```
x = [1 2 3 4 5 6];
y = [2 6 8 7 8 5];
stem(x,y);
title('\bfExample of a Stem Plot');
xlabel('\bf\itx');
ylabel('\bf\ity');
axis([0 7 0 10]);
```

针头图，条形图，罗盘图可以调用 stair，bar，barh 和 compass 命令来创建，代码类似于上面的语句。这些图象的具体细节，例如它们选择性参数，可以通过 **MATLAB** 在线帮助系统得到。

函数 pie 与前面其他的画图有所不同。为了创建一个饼图，程序员把数组 x 传递给函数，函数计算出每一个元素占全部元素和的百分比。例如，如果数组 x 是[1 2 3 4]，那么 pie 函数将会计算出第一个元素 1 占全部元素和的 10%，第二个元素占 20%等等。这个函数将会占这个百分比画出相应的饼图。

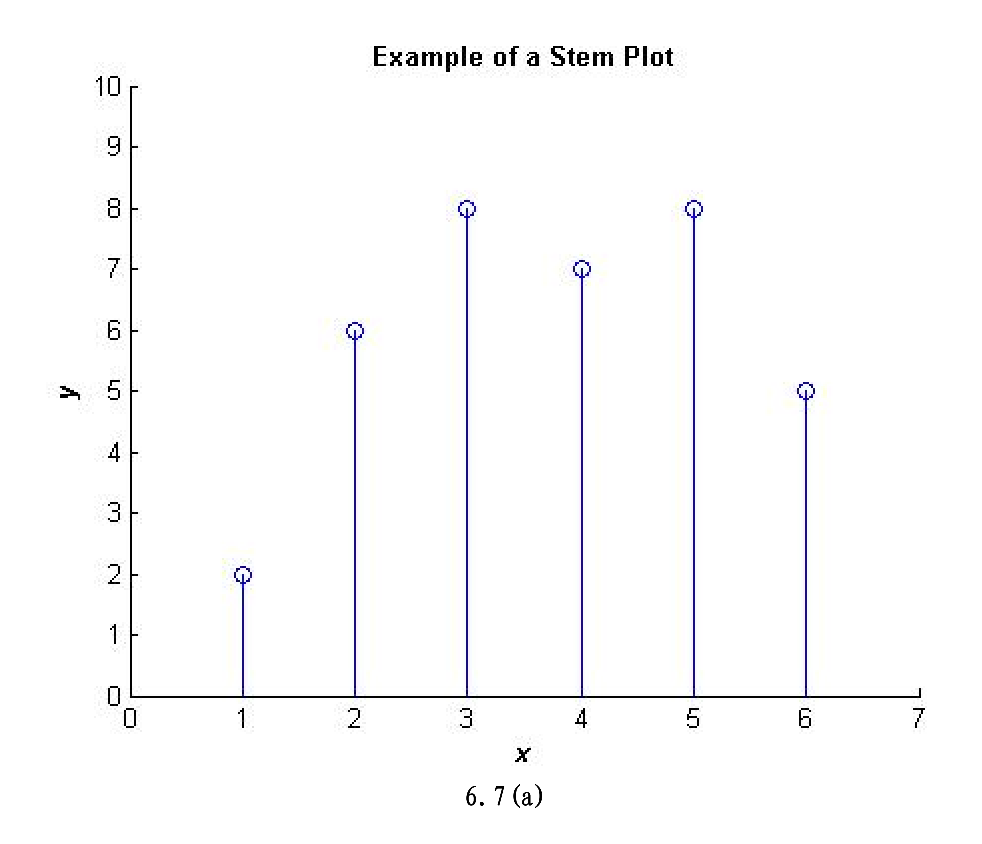
函数 pie 也支持选择性参数，它是 eplode。如果存在的话，explode 是一个逻辑数组，包含元素 1 和 0。如果 explode 的值为 1，那么它对应的扇区就从整体中分离出来。下面的代码将会创建出饼图 6.7e。注意下面的第二个扇区分离出来的。

```
data = [10 37 5 6 6];
explode = [0 1 0 0 0];
```

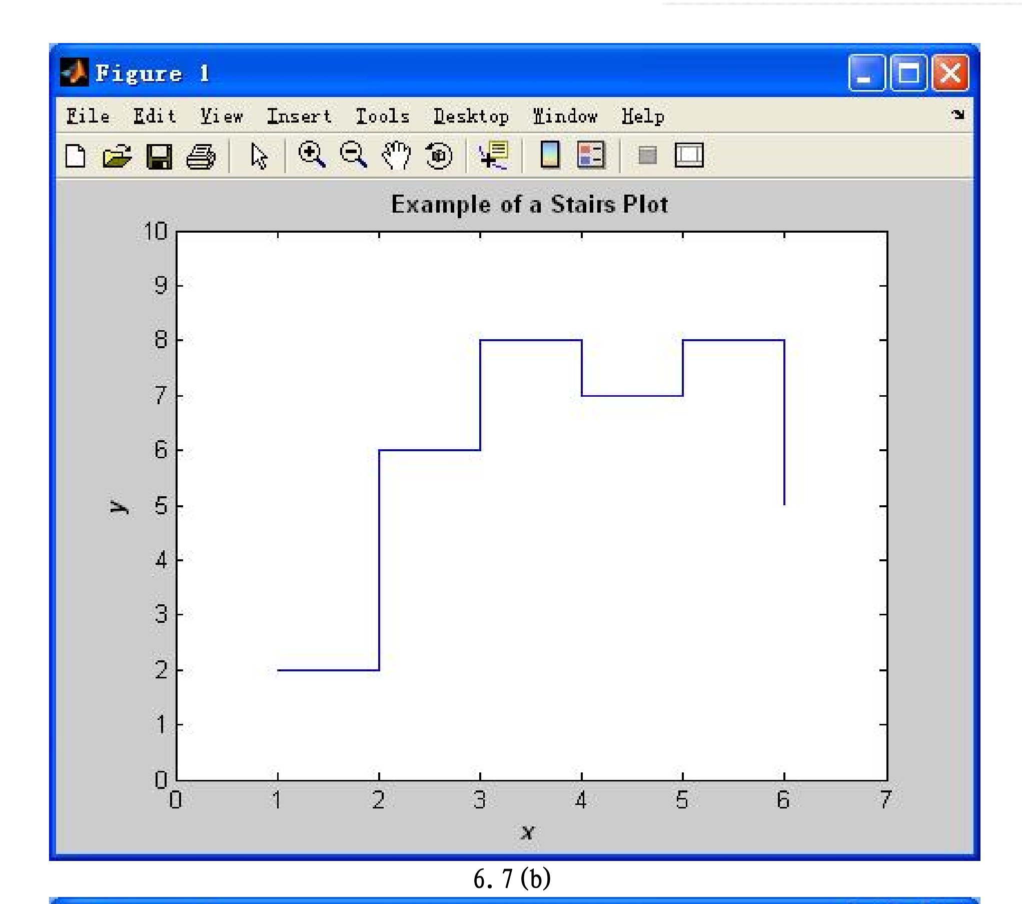
```
pie(data, explode);
title('\bfExample of a Pie Plot');
legend('One','Two','Three','Four','Five');
```

表 6.3 附加的二维作图类型

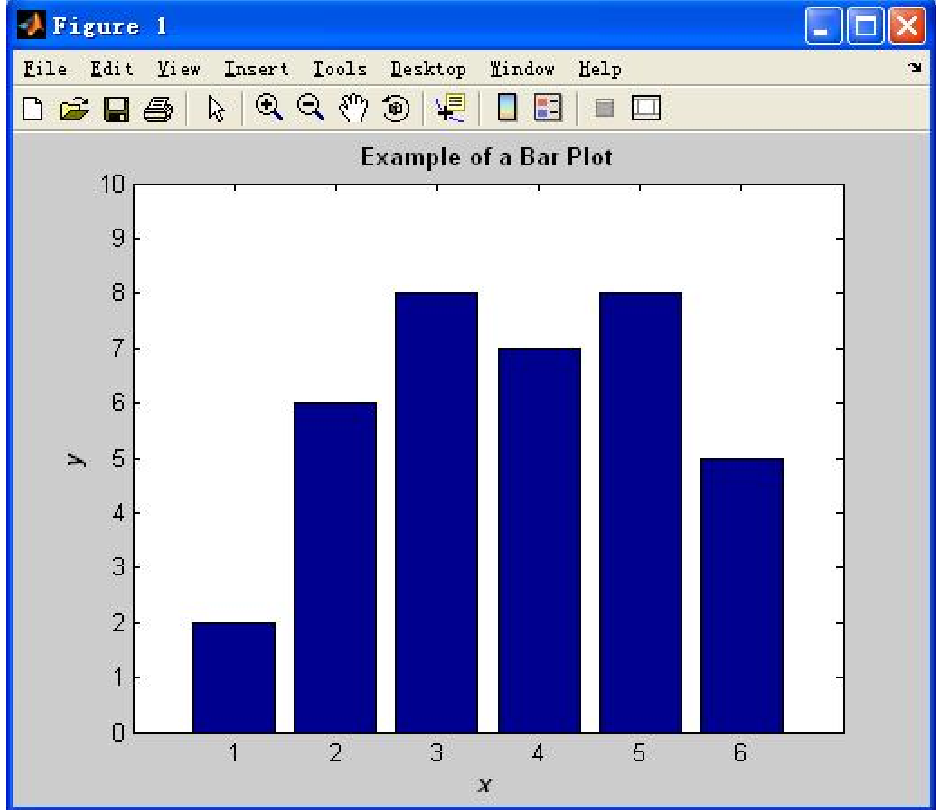
函数	描述
bar(x, y)	这个函数用于创建一个水平的条形图，x 代表第一个 X 轴的取值，y 代表对应于 Y 的取值
barh(x, y)	这个函数用于创建一个竖直的条形图，x 代表第一个 X 轴的取值，y 代表对应于 Y 的取值
compass(x, y)	这个函数用于创建一个极坐标图，它的每一个值都用箭头表示，从原点指向（x，y），注意：（x，y）是直角坐标系中的坐标。
pie(x) pie(x, explode)	这个函数用来创建一个饼状图，x 代表占总数的百分数。explode 用来判断是否还有剩余的百分数
stairs(x, y)	用来创建一个阶梯图，每一个阶梯的中心为点(x, y)
stem(x, y)	这个函数可以创建一个针头图，它的取值为(x,y)



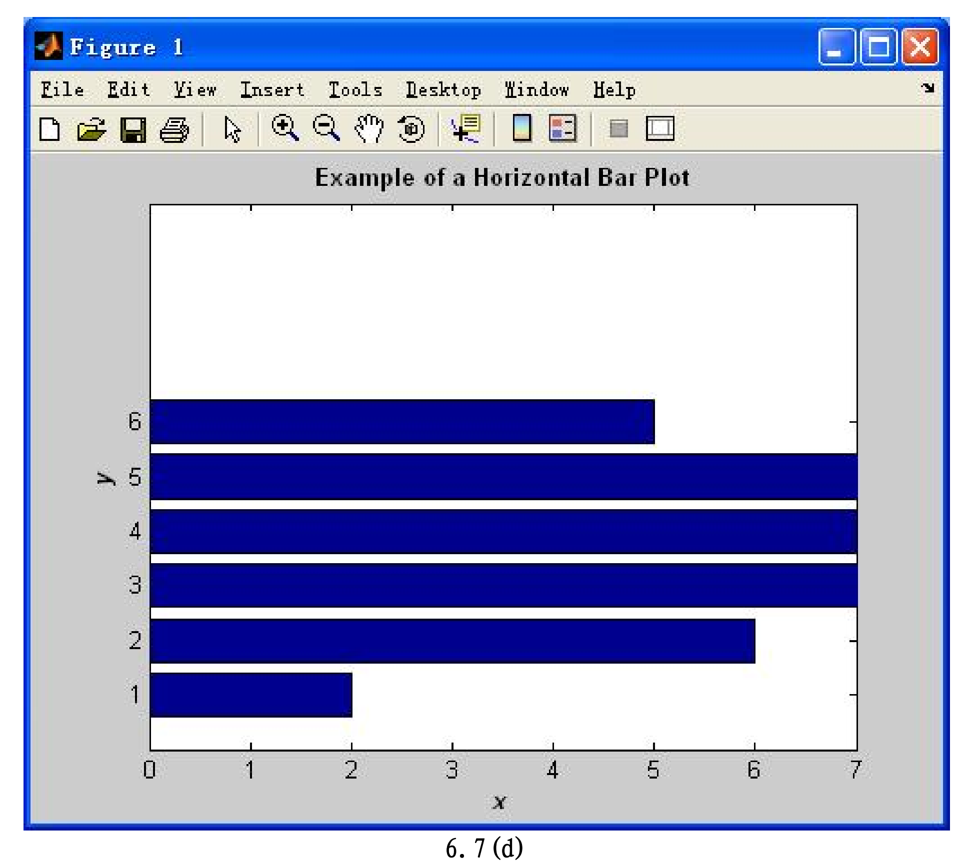
6.7 (a)


Figure 1
File Edit View Insert Tools Desktop Window Help
Example of a Stairs Plot
y
x
0
1
2
3
4
5
6
7
8
9
10

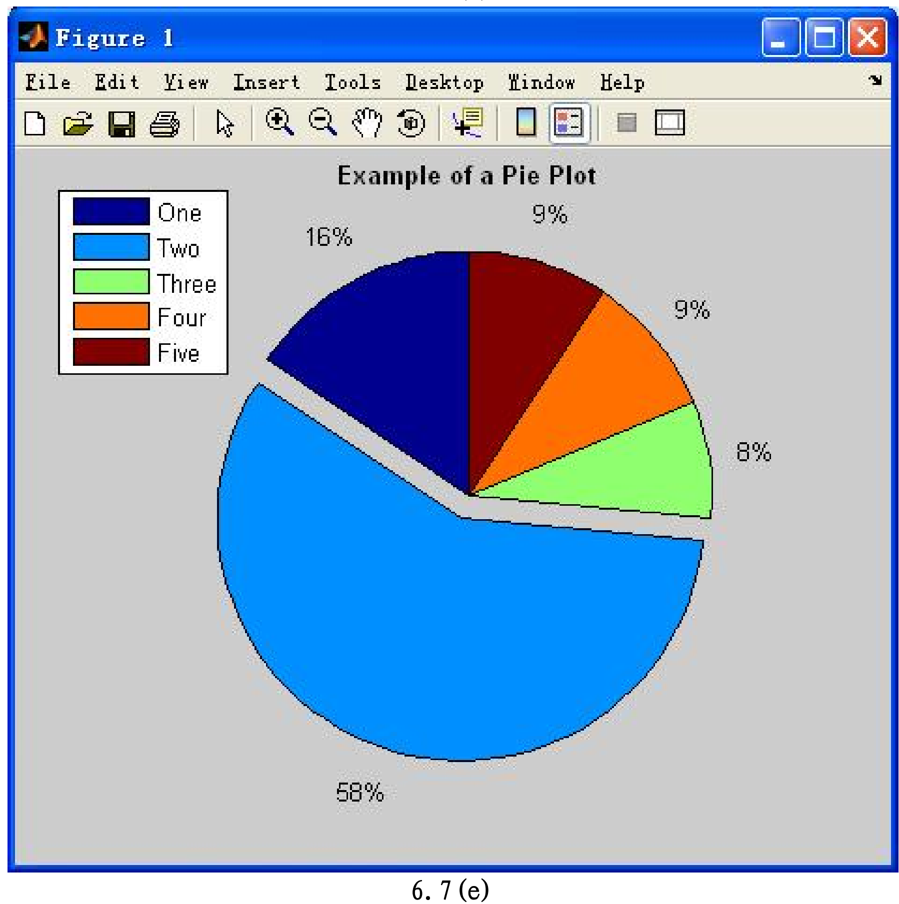
6. 7 (b)


Figure 1
File Edit View Insert Tools Desktop Window Help
Example of a Bar Plot
y
x
0
1
2
3
4
5
6
7
8
9
10

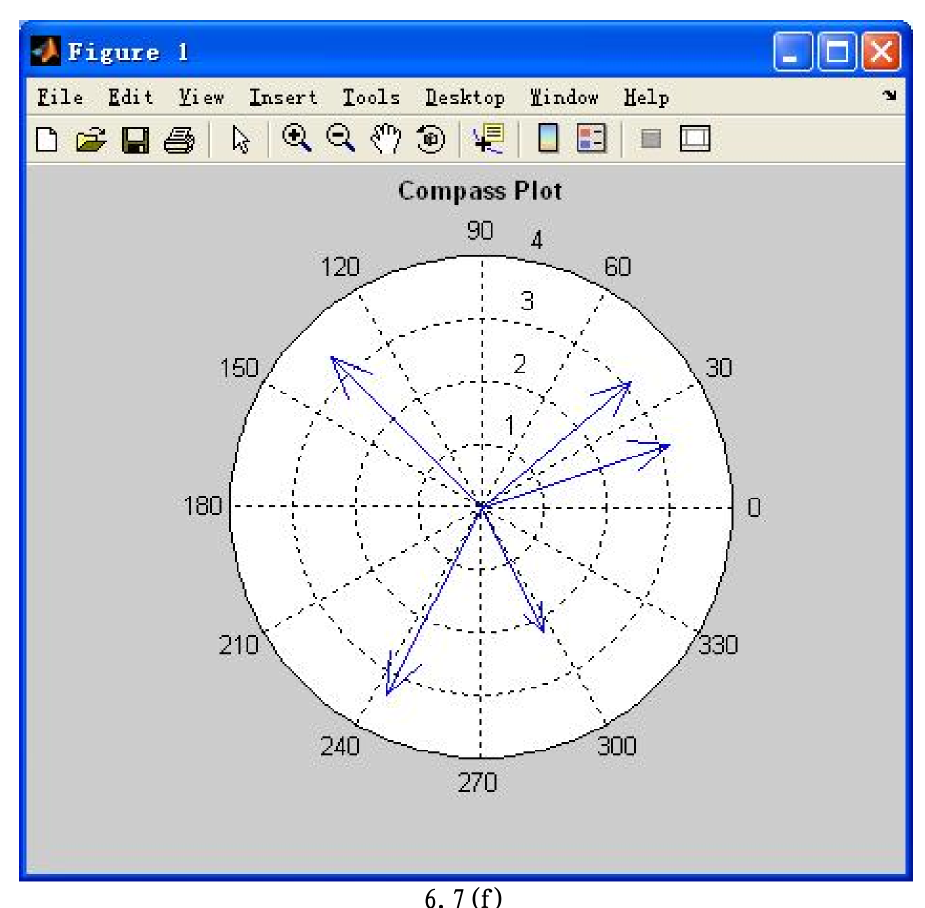
6. 7 (c)


Figure 1
File Edit View Insert Tools Desktop Window Help
Example of a Horizontal Bar Plot
y
1
2
3
4
5
6
0
1
2
3
4
5
6
7
x

6.7(d)


Figure 1
File Edit View Insert Tools Desktop Window Help
Example of a Pie Plot
One
Two
Three
Four
Five
16%
9%
9%
8%
58%

6.7(e)



6.7(f)

图 6.7 各种图象

6.4.2 作图函数

在前面的所有作图，我们必须创建数组，并把这些数组传递给作图函数。**MATLAB** 提供了两个函数可以直接作出图象，而不需要创建中间数据数组。它们是函数数 ezplot 和 fplot。

ezplot 调用函数的形式如下

```
ezplot( fun);
ezplot( fun, [xmin xmax]);
ezplot( fun, [xmin xmax], figure);
```

其中，fun 代表一个字符串，用来表示要画的基本表达式。选择性参数[xmin，xmax]指定自变量的取值范围。如果它不存在的话，函数自变量的范围从-2π到2π。选择性参数图来用指定图象数。

例如，下面语句打印出函数 f(x)=sinx/x，x 的取值范围在-4π 到 4π，输出图象如图 6.8 所示。

```
ezplot('sin(x)/x',[-4*pi 4*pi]);
title('Plot of sinx/x');
grid on;
```

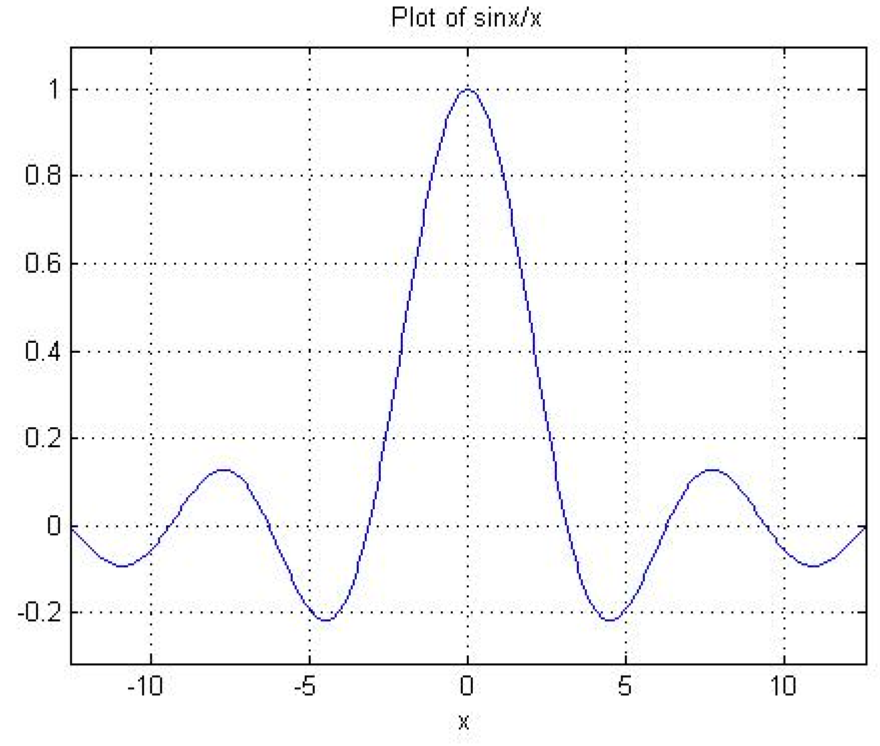


图 6.8 函数 sin(x)/x 的图象

函数 fplot 与 ezplot 相类似，但更加精确。前两个参数与函数 ezplot 中的相同，但是函数 fplot 还有其他优点。

1. 函数 fplot 是适应性的，它意味着在自变量范围内函数突然变化显示更多的点。
2. 函数 fplot 支持 T_EX 命令，用来指定坐标图的标题和坐标轴标签，而函数 ezplot 则不能。

在一般情况下，在画函数图象时，你应当使用函数 fplot。

函数 ezplot 和 fplot 是第五章中“函数的函数”的具体例子。

好的编程习惯

使用 fplot 函数直接打印函数，而不需创建中间数据数据。

6.4.3 柱状图

柱状图用来显示一系列数据数值的分布。为了创建一个柱状图，在一系列数值中范围被平均划分，并确定某一个范围中数值的个数。并把这个数目通过函数画出图来。

标准的 **MATLAB** 柱状图函数应为 hist。函数的形式如下：

```
hist (y)
hist(y, nbins)
his(y, x);
[n, xout] =hist(y, ...)
```

第一个函数创建并画出一个 10 等分的柱状图，而第二种形式创建的是以 nbins 为宽度的柱状图。第三个函数允许用户用数组 x 指定柱状图中长条的中心。这个函数创建柱状图长条都是以数组中的元素为中心的。这三种形式均能创建柱状图。这个函数的最后一种形式创建了一个柱状图并返回了一个数组 xcout，在数组 n 中的每一长条的数目，而实际上并没有创建一个图象。

例如，下面的语句创建一个包含有 1000 个符合正分布的随机数数组并产生了一个取值范围 15 等分的柱状图。产生的图象如图 6.9 所示。

```
y = randn(10000, 1);
hist(y, 15);
```

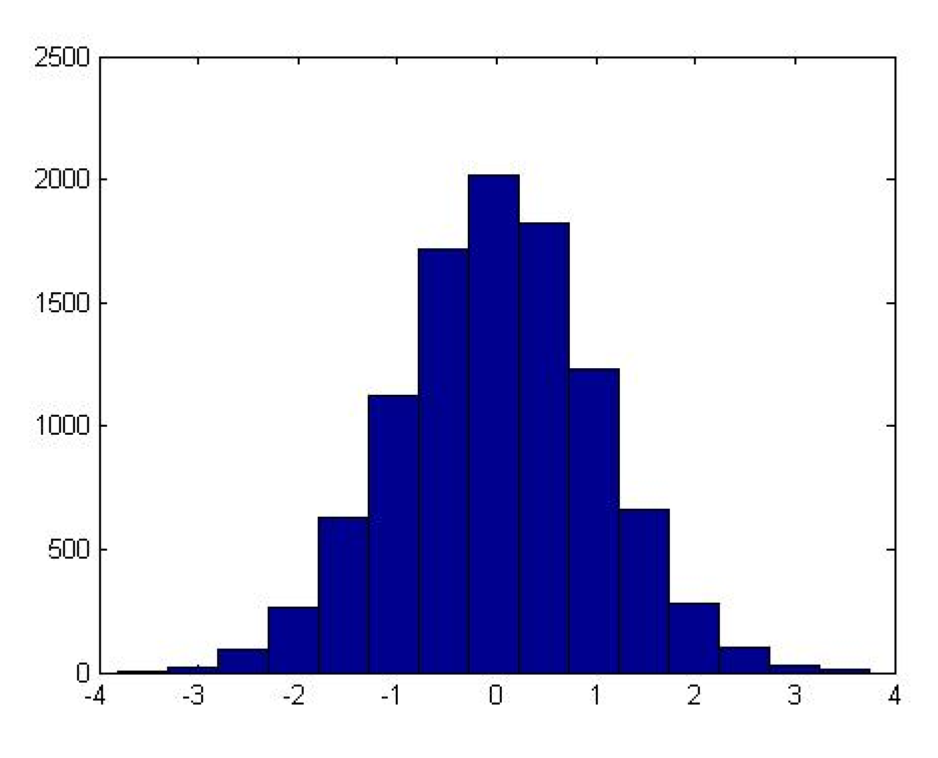


图 6.9 柱状图

MATLAB 提供了函数 rose 用来创建极坐标系中的柱状图。对于研究角度的分布非常有用。你将会在章末的练习中用到。

6.5 三维作图

MATLAB 中包括了丰富的三维图象，用来显示各式各样的数据。在一般情况下，三维图象常用于显示以下两类数据。

1. 两个变量是同一自变量的函数，当你希望显示自变量重要性时，你可以用三维作图表示
2. 一个变量是另外两个变量的函数

6.5.1 三维曲线作图

我们可以用 plot3 函数进行三维曲线的作图。这个函数与二维 plot 函数非常相似，除了每一个点用 x，y，z 表示，而不用 x，y 表示。它的最简单的函数为

```
plot(x, y, z);
```

其中 x，y，z 是等大小的数组，它们组成了这个点的 3 维坐标。函数 plot3 提供了和 plot 函数相同的线型，大小和颜色，你直接利用前面学到的知识，画出一定的图形。作为一个三维曲线的例子，考虑下面的函数

$$x(t) = e^{-0.2t}\cos 2t$$
$$y(t) = e^{-0.2t}\sin 2t$$

这些函数表示在二维机械系统振动衰退情况，所以 x，y 代表在时刻 t 系统的位置。注意 x，y 有一相同的自变量 t。

我们可以创建一系列（x，y）并用二维函数 plot 画出（x，y）的图象，但是我们如果这样作了，时间的重要性就得不到体现。下面的语句创建了物体位置的一个二维图象，如图 6.10a。这个图不可能告诉我们振动变化的快慢。

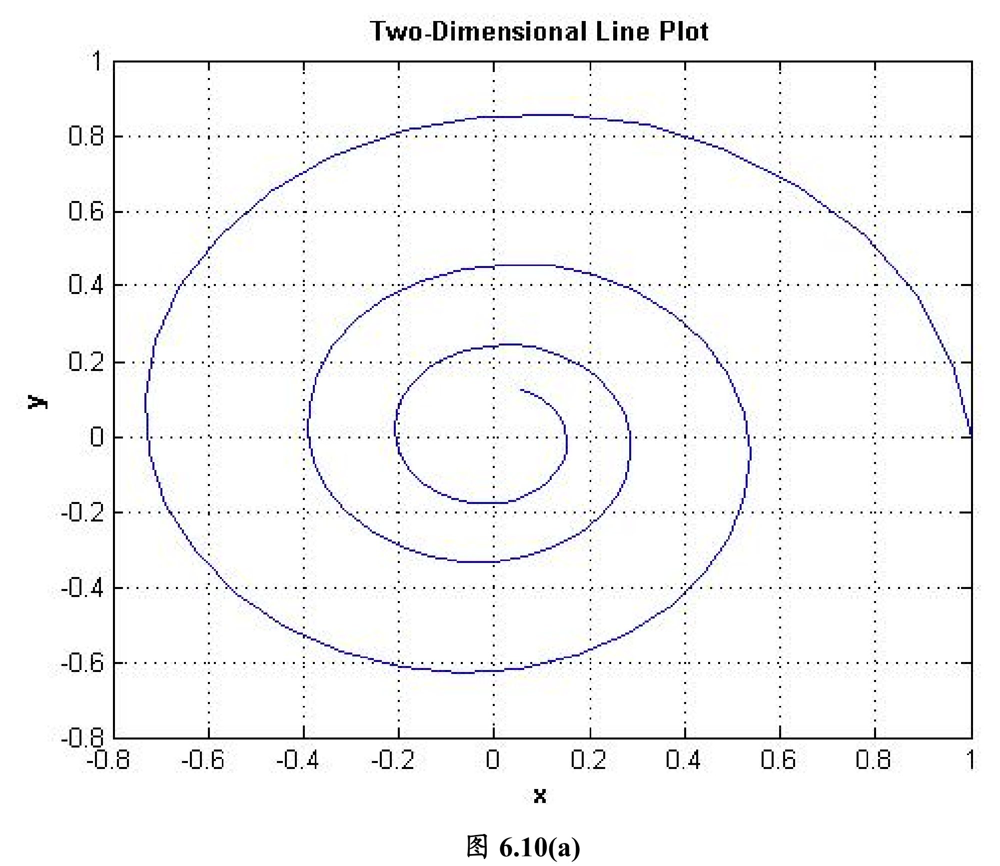


图 6.10(a)

我们可以利用 plot3 函数画出时间物体位置的三维图象。下面的语句将会创造 6.11 的三维图象。

```
t = 0:0.1:10;
x = exp(-0.2*t) .* cos(2*t);
y = exp(-0.2*t) .* sin(2*t);
plot3(x,y,t);
title('\bfThree-Dimensional Line Plot');
xlabel('\bfx');
ylabel('\bfy');
zlabel('\bfTime');
axis square;
grid on;
```

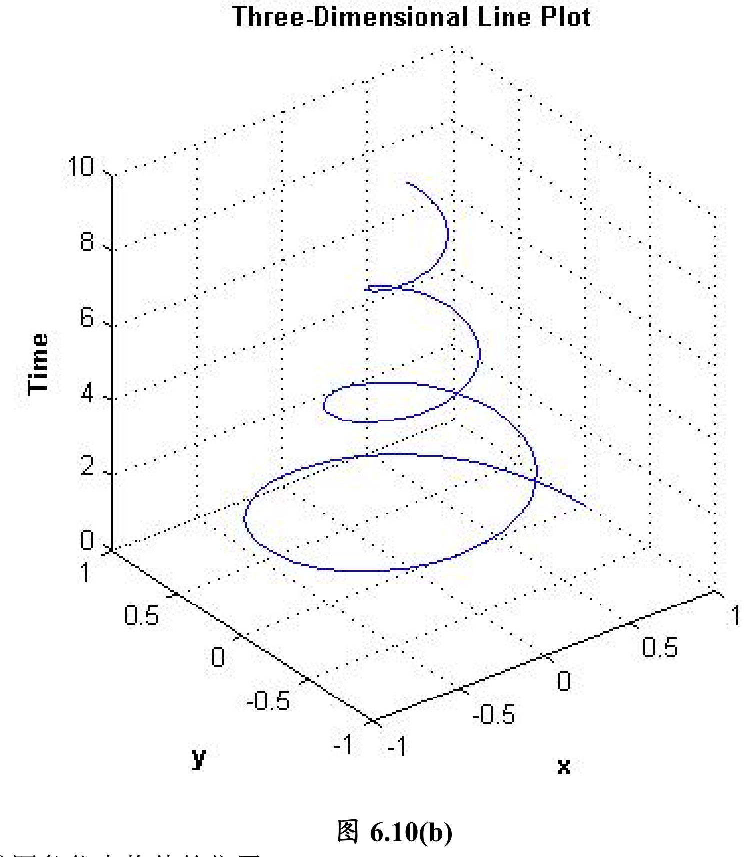


图 6.10(b)

(a)用二维图象代表物体的位置

(b)用三维图象来表示指定时间内的物体的位置产生的图象如图 6.0b 所示。注意这个图象显示了时间的独立性。

6.5.2 三维表面，网格，等高线图象

表面，网格，等高线图象是非常简单的方法来表示两变量的函数。例如，东西（x）南北（y）点上的温度。任何两变量函数的值都能用表面，网格，或等高线图象表示。有更多作图类型出现表 6。4 中，每一个图象的例子显示在图 6。11 中。

利用其中一个函数画图，用户必须创建三个等大小的数组。这个三个数组必须包括要画的每一点的 x，y，z 值。举一个简单的例子，假设我们要 4 个点（-1，-1，1），（1，-1，2），（-1，1，1）和（1，1，0），为了画出这 4 个点，我们必须创建三个数组

$$x=\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}, y=\begin{bmatrix}-1 & -1\\ 1 & 1\end{bmatrix} \text{和 } z=\begin{bmatrix}1 & 2\\ 1 & 0\end{bmatrix}$$

数组 x 包括要画得每一点的 x 值，数组 y 包括要画得每一点的 y 值，数组 z 包括要画得每一点的 z 值。这些数组被传递到画图函数。

表 6.4 表面，网格，等高线图象函数

函数	描述
mesh(x, y, z)	这个函数创建一个三维网格图象。数组 x 包括要画得每一点的 x 值，数组 y 包括要画得每一点的 y 值，数组 z 包括要画得每一点的 z 值。
surf(x, y, z)	这个函数创建一个三维表面图象。x，y，z 代表的意义与上式相同。
contour(x, y, z)	这个函数创建一个三维等高线图象。x，y，z 代表的意义与上式相同。

MATLAB 函数 meshgrid 使函数图象数组 x，y 的创建变得十分容易。这些函数的形式

为

```
[x y] = meshgrid(xstart:xinc:xend, ystart:yinc:yend);
```

xstart:xinc:xend 指定 x 值，ystart:yinc:yend 指定 y 值。

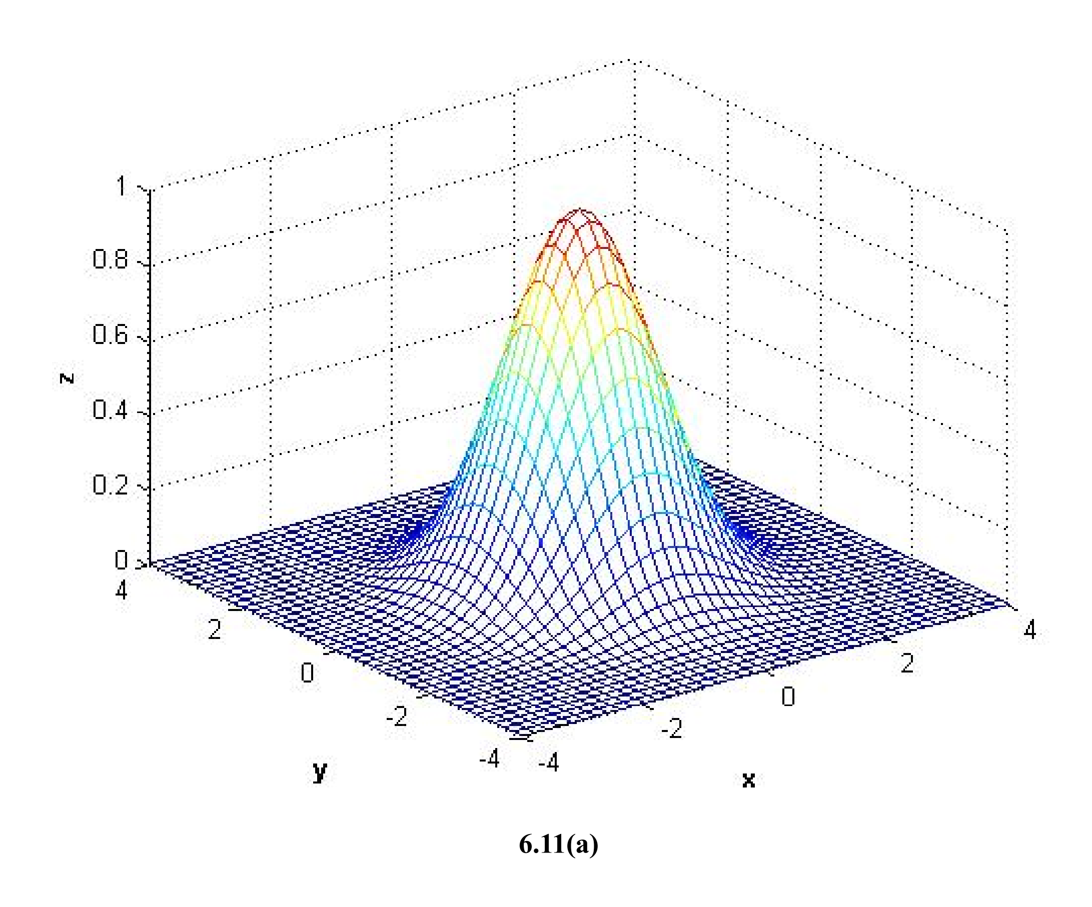
为了创建一个图象，我们要应用 meshgrid 来创建 x，y 的值，并通过函数计算（x，y）相对应的值。最后我们调用函数 mesh，surf 或 contour 来创建图象。例如，我们考虑下面函数的网格图象。

$$z(x, y) = e^{-0.5(x^2 + y^2)} \qquad (6.12)$$

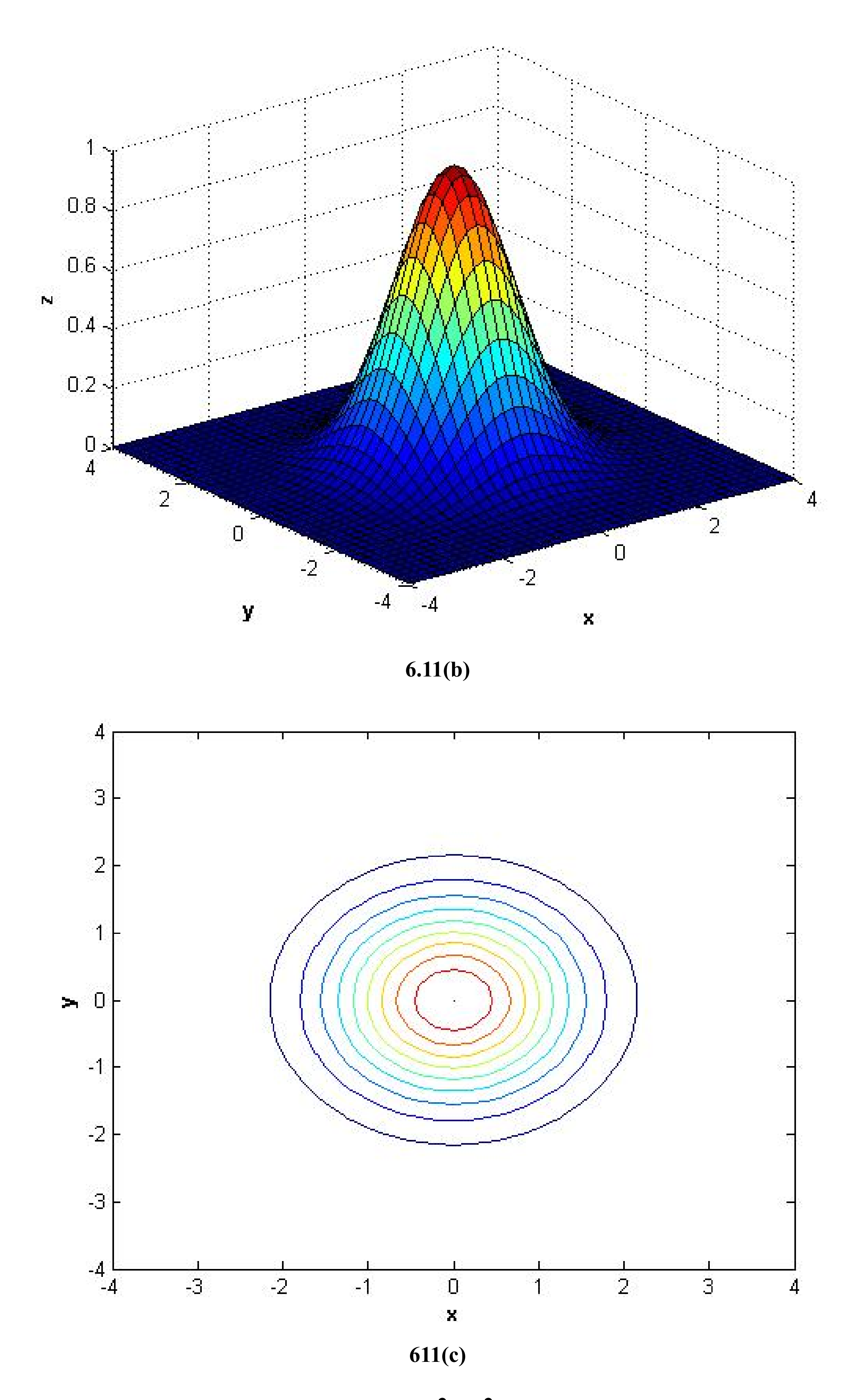
x，y 的取值分别为[-4, 4]，[-4,4]。下面语句将会创建这个图象，如图 6.11a 所示

```
[x,y] = meshgrid(-4:0.2:4,-4:0.2:4);
z = exp(-0.5*(x.^2+y.^2));
mesh(x,y,z);
xlabel('\bfx');
ylabel('\bfy');
zlabel('\bfz');
```

表面，等高线图象可以类似于 mesh 函数创建。



6.11(a)



6.11(b)

611(c)

图 6.11 (a)(b)(c)分别代表函数$z(x, y) = e^{-0.5(x^2 + y^2)}$网格图，表面图和等高线图。

6.6 总结

MATLAB 支持复数，作为 double 数据类型的扩展。复数用 i 和 j 定义，它们都被预定义为$\sqrt{1}$ 。你可以直接应用复数进行运算，但要注意关系运算符只对复数的实部进行比较，而不对它的模进行比较。当你进行复数运算这是一个常见错误。

字符串函数是进行字符串操作。字符串是 char 型数组。这些函数允许用户对字符串进行各种各样的操作，例如连接，比较，替换，查找，大小写转换，数字与字符串之间的转换。

多维数组是指超过两维的数组。它们可用创建一维，二维数组的方法进行创建。多维数组用于解决自然界的一些问题。

MATLAB 中包含了许多的二维三维的作图方法，在本章中我们向大家介绍了针头图（stem Plots），阶梯图（stair plots），条形图，饼图（pie plots），罗盘图（compass plots），三维表面，网格，等高线图象。

6.6.1 好的编程习惯总结

下面是要我们遵守的指导原则

1. 用 char 函数创建二维字符数组，我们就不用担心每一行的长度不相同了。
2. 我们可以利用多维数组来解决自然界的多变量问题，如空气动力学和流体力学。
3. 使用 fplot 函数直接打印函数，而不需创建中间数据数据。

6.6.2 MATLAB 函数与命令总结

函数	描述
char	(1)把数字转化为相应的字符值
	(2)把二维数组转化相应的字符串
double	把字符转化为相应的 double 值
blanks	创建一个由空格组成的字符串
deblanks	去除字符串末端的空格
strcat	连接字符串
strvcat	竖直地连接字符串
strcmp	如果两字符串相等，那么函数将会返回 1
stricmp	忽略大小写如果两字符串相等，那么函数将会返回 1
strncmp	如果两字符串的前 n 个字母相等，那么函数将会返回 1
strncmpi	忽略大小，如果两字符串的前 n 个字母相同，那么数将会返回 1
findstr	在一个字符串中寻找另一个字符串
strfind	在一个字符串中寻找另一个字符串（版本 6。1 或以后的版本）
strjust	对齐字符串
strmatch	找字符串的区配
strrep	用一个字符串去替代另一个字符串
strtok	查找一字符串
upper	把字符串的所有字符转化为大写
lower	把字符串的所有字符转化为小写
int2str	把整数转化为相应的字符串形式
num2str	把数字转化为相应的字符串形式

函数	描述
mat2str	把矩阵转化为相应的字符串形式
sprintf	对一字符串进行格式化输出
str2double	把字符串转化相应的 double 型数据
str2num	把字符转化成数字
sscanf	从字符串中读取格式化数据
hex2num	把 IEEE 十六进制字符型型数据转化为 double 形数据
hex2dec	把十六制字符串转化为相应的十进制整数
dec2hex	把十进制数转化为相应的十六制字符串
bin2dec	把二进制字符串转化为相应的十进制整数
base2dec	把 baseb 转化为相应的十进制数据
dec2base	把十进制转化为相应的 baseb
bar(x, y)	这个函数用于创建一个水平的条形图，x 代表第一个 X 轴的取值，y 代表对应于 Y 的取值
barh(x, y)	这个函数用于创建一个竖直的条形图，x 代表第一个 X 轴的取值，y 代表对应于 Y 的取值
compass(x, y)	这个函数用于创建一个极坐标图，它的每一个值都用箭头表示，从原点指向（x，y），注意：（x，y）是直角坐标系中的坐标。
pie(x) pie(x, explode)	这个函数用来创建一个饼状图，x 代表占总数的百分数。explode 用来判断是否还有剩余的百分数
stairs(x, y)	用来创建一个阶梯图，每一个阶梯的中心为点(x, y)
stem(x, y)	这个函数可以创建一个针头图，它的取值为(x,y)

6.7 练习

6.1 图 6.12 显示的是一个 *RLC* 电路，它的激励产生一个 $120\angle 0^{o}$V 的电压。这个电路中的感抗为 $Z_L=j*2\pi fL$，其中 j 为$\sqrt{1}$ ，f 是电压的频率，单位为 Hz。这个电路的容抗为 $Z_C = -j * \frac{1}{2\pi fC}$ ，C 为电容。假设 R=100Ω，L=0.1mH，C=0.25nF。

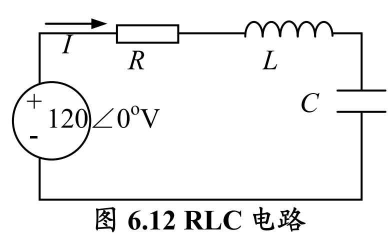


图 6.12 RLC 电路

根据基尔霍夫电压定律我们可以得到电流$I = \dfrac{120\angle 0^{o}\text{V}}{R + j2\pi fL - j\dfrac{1}{2\pi fC}}$

a. 计算并画出以频率为自变量的电流模函数的图象，要求频率从 100KHz 到 10MHz 变化。画出图要求有两个，一个是线性图，一个是对数线性图。保证包括标题和坐标标签。

b. 计算并画出以频率为自变量的电流相角函数的图象。要求频率从 100KHz 到 10MHz 变化。画出图要求有两个，一个是线性图，一个是对数线性图。保证包括标题和坐标标签。

c.画出以频率为自变量的电流相角，模函数的图象。要求在同一图象窗口内画出两个图象。用对数线性标度。

6.2 编一个函数 to_polar，可以接受一个复数，并返回两个输出参数——复数 C 的模 mag 和角度 theta。输出角度的单位为度。

6.3 编写一个函数to_complex，可以接受两个输入参数——复数C的模mag和角度theta，

返回复数 C。

6.4 在一个稳定的交变电路中，通过一中性电路元件时符合欧姆定律为

$$V=IZ \quad (6.13)$$

V 代表这个元件两端的电压，I 代表通过这个器件的电流，Z 代表这个元件的阻抗。注意这个三个数都是复数。而这些数一般用模与相角的形式表示。例如，电压可能为 $V=120\angle 30^o$。

编写一个程序，读取负载两端的电压，和负载的阻抗，并计算出电流值。输入应为模与相角，输出也应为这种格式。用练习 6.3 和 6.2 中的程序进行格式转换。

6.5 编写一个程序，接受一个复数 C 并在笛卡尔坐标系中用圆圈画出相应的点。图象应包括 x，y 轴，并要求生成一个由原点指向 C 的向量。

6.6 调用函数 plot(t，v)画出数学函数 $v(t)=10e^{(-0.2+j\pi)t}$ 在 $0\leqslant t\leqslant 10$ 时图象。这个图象是什么？

6.7 调用函数 plot(v)画出数学函数 $v(t)=10e^{(-0.2+j\pi)t}$ 在 $0\leqslant t\leqslant 10$ 时图象。这次图象是什么？

6.8 创建一个数学函数在 $0\leqslant t\leqslant 10$ 时的极坐标图。

6.9 调用函数 plot(t，v)画出数学函数 $v(t)=10e^{(-0.2+j\pi)t}$ 在 $0\leqslant t\leqslant 10$ 时图象，图象的每一维分别是函数的实部，虚部和时间。

6.10 欧拉公式。公式定义如下

$$e^{i\theta} = \cos\theta+i\sin\theta \quad (6.14)$$

创建一个二维图象，θ 的取值 0 到 2π 之间。

创建一个三维图象，θ 的取值 0 到 2π 之间。

6.11 创建函数 $z=e^{x+iy}$ 在 $-1\leqslant x\leqslant 1$ 和 $-2\pi\leqslant y\leqslant 2\pi$ 内的三维网图，表面图和等高线图。

6.12 编写一个程序，从用户接受一个字符串，并确定用户指定的字母出现在字符串多少次（提示：在 **MATLAB** 帮助工作台中查找 input 函数的“s”参数选项。）

6.13 修改上面的程序，确定用户指定的字母出现在字符串多少次，忽略字母的大小写。

6.14 编写一个程序，用 input 函数接受一个字符串，并对这些字符串分解成各种符号，然后对这些符号进行按升序排序，并把他们打印出来。

6.15 编写一个程序，用 input 函数接受一系列字符串，并对这些字符串按升序进行排序，并把他们打印出来。

6.16 编写一个程序，用 input 函数接受一系列字符串，并对这些字符串按升序进行排序，忽略大小写，并把他们打印出来。

6.17 **MATLAB** 中包含两个函数 upper 和 lower，它分别把字符串转化为大字和小写。创建一个新的函数 caps，让字符串每个单词的第一个字母大写，其余的为小写。

6.18 画出函数 $y=e^x\sin x$ 的针头图，阶梯图，条形图，罗盘图。确保所有图象中都有标题和坐标轴标签。

6.19 假设 George，Sam，Betty，Charlie 和 Suzie 去送别一位同事，他们为买礼物分别花了\$5，\$10，\$7，\$15。创建一个饼图，Sam 花得钱占总数的百分比为多少？

6.20 用函数 fplot 画出函数 $f(x)=\dfrac{1}{\sqrt{x}}$ ，x 在 $0.1\leqslant x\leqslant 10.0$ 中的图象。

第七章 稀疏矩阵 单元阵列 结构

在本章中我们要学习三种数据类型：稀疏矩阵，单元阵列和结构。稀疏矩阵是矩阵的一种特殊形式，在这个矩阵中只对非零元素分配内存。单元阵列也是一种矩阵，它的每一个元素可以是 **MATLAB** 任何一种数据类型。它们广泛应用于 **MATLAB** 用户图象界面（GUI）函数。最后，结构提供了一种在单个变量中存储了不同类型的数据的方法，在这个变量中的每一个数据项目都有一个独立的名字。

7.1 稀疏矩阵

我们在第二章中已经学过了普通的 **MATLAB** 数组。当一个普通的数组被声明后，**MATLAB** 将会为每一个数组元素分配内存。例如函数 a = eye(10)要创建了 100 个元素，按 10×10 的结构分配，对角线上的元素均为 1，其余的元素为 0。注意这些数组其中的 90 个元素为 0。这个包含有一百个元素的矩阵，只有 10 个元素包含非零值。这是稀疏矩阵或稀疏数组的一个例子。稀疏矩阵是指一个很大的矩阵，且大多数的元素为 0。

```
>> a=2*eye(10)
a =
     2     0     0     0     0     0     0     0     0     0
     0     2     0     0     0     0     0     0     0     0
     0     0     2     0     0     0     0     0     0     0
     0     0     0     2     0     0     0     0     0     0
     0     0     0     0     2     0     0     0     0     0
     0     0     0     0     0     2     0     0     0     0
     0     0     0     0     0     0     2     0     0     0
     0     0     0     0     0     0     0     2     0     0
     0     0     0     0     0     0     0     0     2     0
     0     0     0     0     0     0     0     0     0     2
```

现在假如我们要创建一个 10×10 的矩阵，定义如下

```
b =
     1     0     0     0     0     0     0     0     0     0
     0     2     0     0     0     0     0     0     0     0
     0     0     2     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     5     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     1
```

若 a，b 两矩阵相乘得到的结果为

```
>> c = a * b
c =
     2     0     0     0     0     0     0     0     0     0
     0     4     0     0     0     0     0     0     0     0
     0     0     4     0     0     0     0     0     0     0
     0     0     0     2     0     0     0     0     0     0
     0     0     0     0    10     0     0     0     0     0
```

```
0    0    0    0    0    2    0    0    0    0
0    0    0    0    0    0    2    0    0    0
0    0    0    0    0    0    0    2    0    0
0    0    0    0    0    0    0    0    2    0
0    0    0    0    0    0    0    0    0    2
```

这两个稀疏矩阵相乘需要 1900 次相加和相乘，但是在大多数时候相加和相乘的结果为 0，所以我们做了许多的无用功。这个问题会随着矩阵大小的增大而变得非常的严重。例如，假设我们要产生两个 200×200 的稀疏矩阵，如下所示

```
a = 5 * eye(200);
b = 3 * eye(200);
```

每一个矩阵有 20000 个元素，其中 19800 个元素是 0。进一步说，对这两个矩阵相乘需要 7980000 次加法和乘法。

我们可以看出对大规模稀疏矩阵进行存储和运算（其中大部分为 0）是对内存空间和 cpu 资源的极大浪费。不巧的是，在现实中的许多问题都需要稀疏矩阵，我们需要一些有效的方示来解决这个问题。

大规模供电系统是现实世界中涉及到稀疏矩阵一个极好的例子。大规模供电系统可以包括上千条或更多的母线，用来产生，传输，分配电能到子电站。如果我们想知道这个系统的电压，电流和功率，我们必须首先知道每一条母线的电压。如果这个系统含有一千个母线，这就要用到含有 1000 个未知数的联立方程组，包括一个方程，也就是说我们要创建含有 1000000 个元素的矩阵。解出这个矩阵，需要上百万次的浮点运算。

但是，在这个系统中，一条母线平均只它的三条母线相连，而在这个矩阵中每一行其他的 996 个元素将为 0，所以在这个矩阵的加法和乘法运算中将会产生 0。如果在求解的过程中这些 0 可以忽略，那么这个电力系统的电压和电流计算将变得简单而高效。

7.1.1 sparse 数据类型

在 **MATLAB** 中有一个专门的数据类型，用来对稀疏进行运算。sparse 数据类型不同于 doulbe 数据，它在内存中只存储非零元素。实际上，sparse 数据类型只存储每一个非零元素的三个值：元素值，元素的行号和列号。尽管非零元素这三个值必须存储在这内存，但相对于存储稀疏矩阵的所有元素来说要简单高效得多。

我们用 10×10 的方阵来说明稀疏矩阵的应用。

```
>> a = eye(10)

a =

    1    0    0    0    0    0    0    0    0    0
    0    1    0    0    0    0    0    0    0    0
    0    0    1    0    0    0    0    0    0    0
    0    0    0    1    0    0    0    0    0    0
    0    0    0    0    1    0    0    0    0    0
    0    0    0    0    0    1    0    0    0    0
    0    0    0    0    0    0    1    0    0    0
    0    0    0    0    0    0    0    1    0    0
    0    0    0    0    0    0    0    0    1    0
    0    0    0    0    0    0    0    0    0    1
```

如果这个矩阵被转化为稀疏矩阵，它的结果是

```
>> as = sparse(a)
as =
   (1,1)        1
   (2,2)        1
   (3,3)        1
```

```
    (4,4)        1
    (5,5)        1
    (6,6)        1
    (7,7)        1
    (8,8)        1
    (9,9)        1
   (10,10)       1
```

注意在稀疏矩阵存储的是行列地址和这一点所对应的非零数据值。只要一个矩阵的大部分数都是 0，这种方法用来存储数据就是高效的，但是如果非零元素很多的话，那么用占用更多的空间，因为稀疏矩阵需要存储蓄地址。函数 issparse 通常用作检测一个矩阵是否为稀疏矩阵。如果这个矩阵是稀疏的，那么这个函数将会返回 1。

稀疏矩阵的优点可以通过下面的描述体现出来，考虑一个 1000×1000 的矩阵平均每一行只有 4 个非零元素。如果这个矩阵以全矩阵的形式储存，那么它要战胜 8000000 个字节。从另一方面说，如果它转化为一个稀疏矩阵，那么内存的使用将会迅速下降。

7.1.1.1 产生稀疏矩阵

MATLAB 可以通过 sparse 函数把一个全矩阵转化为一个稀疏矩阵，也可以用 **MATLAB** 函数 speye，sprand 和 sprandn 直接产生稀疏矩阵，它们对应的全矩阵为 eye，rand，和 randn。例如，表达式 a = speye(4)将产生一个 4×4 的稀疏矩阵。

```
>> a = speye(4)
a =
    (1,1)        1
    (2,2)        1
    (3,3)        1
    (4,4)        1
```

表达式 b = full(a)把稀疏矩阵转化相应的全矩阵。

```
>> b = full(a)
b =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

7.1.1.2 稀疏矩阵的运算

如果一个矩阵是稀疏的，那么单个元素可以通过简单的赋值语句添加或删除，例如下面的语句产生一个 4×4 的稀疏矩阵，然后把其他的非零元素加入其中。

```
>> a = speye(4)
a =
    (1,1)        1
    (2,2)        1
    (3,3)        1
    (4,4)        1
>> a(2,1) = -2
a =
    (1,1)        1
    (2,1)       -2
    (2,2)        1
    (3,3)        1
```

```
        (4,4)              1
```

MATLAB 允许全矩阵与稀疏的混合运算。它们产生的结果可以是全矩阵也可以是稀疏矩阵，这取决于那种结果更高效。更重要的是，任何的适用全矩阵算法同样地也适合稀疏矩阵。

表 7.1 列出的是一些普通的稀疏矩阵。

表 7.1 普通的 **MATLAB** 稀疏矩阵函数

类别	函数	描述
创建一个稀疏矩阵	speye	创建一个单位稀疏矩阵
	sprand	创建一个稀疏矩阵，元素是符合平均分布的随机数
	sprandn	创建一个稀疏矩阵，元素是普通的随机数
全矩阵和稀疏矩阵的转换函数	sparse	把一个全矩阵转化为一个稀疏矩阵
	full	把一个稀疏矩阵转化为全矩阵
	find	找出矩阵中非零元素和它对应的上下标
对稀疏矩阵进行操作的函数	nnz	非零元素的个数
	nonzeros	返回一个向量，其中的元素为矩阵中非零元素
	spones	用 1 代替矩阵中的非零元素
	spalloc	一个稀疏矩阵所占的内存空间
	issparse	如果是稀疏矩阵就返回 1
	spfun	给矩阵中的非零元素提供函数
	spy	用图象显示稀疏矩阵

例 7.1

用稀疏矩阵解决联立方程组

为了解说明稀疏矩阵在 **MATLAB** 中应用，我们将用全矩阵和稀疏矩阵来解决下面的联立方程组。

$1.0x_1 + 0.0x_2 + 1.0x_3 + 0.0x_4 + 0.0x_5 + 2.0x_6 + 0.0x_7 - 1.0x_8 = 3.0$
$0.0x_1 + 1.0x_2 + 0.0x_3 + 0.4x_4 + 0.0x_5 + 0.0x_6 + 0.0x_7 + 0.0x_8 = 2.0$
$0.5x_1 + 0.0x_2 + 2.0x_3 + 0.0x_4 + 0.0x_5 + 0.0x_6 - 1.0x_7 + 0.0x_8 = -1.5$
$0.0x_1 + 0.0x_2 + 0.0x_3 + 2.0x_4 + 0.0x_5 + 1.0x_6 + 0.0x_7 + 0.0x_8 = 1.0$
$0.0x_1 + 0.0x_2 + 1.0x_3 + 1.0x_4 + 1.0x_5 + 0.0x_6 + 0.0x_7 + 0.0x_8 = -2.0$
$0.0x_1 + 0.0x_2 + 0.0x_3 + 1.0x_4 + 0.0x_5 + 1.0x_6 + 0.0x_7 + 0.0x_8 = 1.0$
$0.5x_1 + 0.0x_2 + 0.0x_3 + 0.0x_4 + 0.0x_5 + 0.0x_6 + 1.0x_7 + 0.0x_8 = 1.0$
$0.0x_1 + 1.0x_2 + 0.0x_3 + 0.0x_4 + 0.0x_5 + 0.0x_6 + 0.0x_7 + 1.0x_8 = 1.0$

答案

为了解决这一问题，我们将创建一个方程系数的全矩阵，并用 sparse 函数把他转化为稀疏矩阵。我们用两种方法解这个方程组，比较它们的结果和所需的内存。

代码如下：

```
% Script file: simul.m
%
% Purpose:
% This program solves a system of 8 linear equations in 8
% unknowns (a*x = b), using both full and sparse matrices.
%
% Record of revisions:
% Date          Programmer          Description of change
% ====          ==========          =====================
% 10/14/98      S. J. Chapman       Original code
%
```

```
% Define variables:
% a            --Coefficients of x (full matrix)
% as           --Coefficients of x (sparse matrix)
% b            --Constant coefficients (full matrix)
% bs           --Constant coefficients (sparse matrix)
% x            --Solution (full matrix)
% xs           --Solution (sparse matrix)
% Define coefficients of the equation a*x = b for
% the full matrix solution.
a = [
1.0 0.0 1.0 0.0 0.0 2.0 0.0 -1.0; ...
0.0 1.0 0.0 0.4 0.0 0.0 0.0 0.0; ...
0.5 0.0 2.0 0.0 0.0 0.0 -1.0 0.0; ...
0.0 0.0 0.0 2.0 0.0 1.0 0.0 0.0; ...
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0; ...
0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0; ...
0.5 0.0 0.0 0.0 0.0 0.0 1.0 0.0; ...
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0
];
b = [ 3.0 2.0 -1.5 1.0 -2.0 1.0 1.0 1.0]';
% Define coefficients of the equation a*x = b for
% the sparse matrix solution.
as = sparse(a);
bs = sparse(b);
% Solve the system both ways
disp ('Full matrix solution:');
x = a\b
disp ('Sparse matrix solution:');
xs = as\bs
% Show workspace
disp('Workspace contents after the solutions:')
whos
```

运行这个程序，结果如下

```
>> simul
Full matrix solution:
x =
    0.5000
    2.0000
   -0.5000
   -0.0000
   -1.5000
    1.0000
    0.7500
   -1.0000
Sparse matrix solution:
xs =
   (1,1)        0.5000
   (2,1)        2.0000
   (3,1)       -0.5000
   (5,1)       -1.5000
   (6,1)        1.0000
   (7,1)        0.7500
   (8,1)       -1.0000
Workspace contents after the solutions:
  Name      Size                    Bytes  Class
  a         8x8                       512  double array
  as        8x8                       276  double array (sparse)
```

```
    b          8x1              64   double array
    bs         8x1             104   double array (sparse)
    x          8x1              64   double array
    xs         8x1              92   double array (sparse)
  Grand total is 115 elements using 1112 bytes
```

两种算法得到了相同的答案。注意用稀疏矩阵产生的结果不包含 x_4，因为它的值为 0。注意 b 的稀疏形式占的内存空间比全矩阵形式还要大。这种情况是因为稀疏矩阵除了元素值之外必须存储它的行号和列号，所以当一个矩阵的大部分元素都是非零元素，用稀疏矩阵将降低运算效率。

7.2 单元阵列(cell array)

单元阵列是 **MATLAB** 中特殊一种数组，它的元素被称为单元(cells)，它可以存储其它类型的 **MATLAB** 数组。例如，一个单元阵列的一个单元可能包含一个实数数组或字符型数组，还可能是复数组(图 7.1 所示)。

在一个编程项目中，一个单元阵列的每一个元素都是一个指针，指向其他的数据结构，而这些数据结构可以是不同的数据类型。单元阵列为选择问题信息提供极好的方示，因为所有信息都聚集在一起，并可以通边一单个名字访问。单元阵列用大括号{}替代小括号来选择和显示单元的内容。这个不同是由于单元的内容用数据结构代替了内容。假设一单元阵列如图 7.2 所示。元素 a(1，1)是数据结构 3×3 的数字数组。a(1，1)的含义为显示这个单元的内容，它是一个数据结构。

cell 1,1	cell 1,2
$\begin{bmatrix} 1 & 3 & -7 \\ 2 & 0 & 6 \\ 0 & 5 & 1 \end{bmatrix}$	'This is a string'
cell 2,1	cell 2,2
$\begin{bmatrix} 3+i4 & -5 \\ -i10 & 3-i4 \end{bmatrix}$	[]

图 7.1 一个单元阵列的一个单元可能包含一个实数数组或字符型数组，还可能是复数组

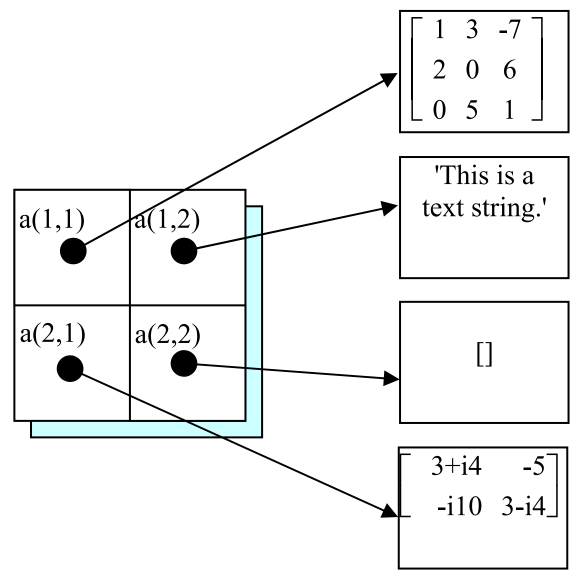


图 7.2 单元阵列中的每一个元素都是指向其他数据结构的指针，指向的数据结构可能都不相同

```
>> a(1,1)
ans =
    [3x3 double]
```

相对地，a{1,1}的含义为显示这个数据结构的内容。

```
>> a{1,1}
ans =
    1    3    -7
    2    0     6
    0    5     1
```

总起来说，标识 a{1,1}反映的是数据结构 a(1,1)内容，而标识 a(1,1)是一个数据结构。

好的编程习惯
当你访问一单元阵列时，不要把()与{}混淆。它们完全不同的运算。

7.2.1 创建单元阵列

创建单元阵列有两种方法

- 用赋值语句
- 用函数 cell 创建

最简单的创建单元阵列的方法是直接把数据结构赋值于独立的单元，一次赋一个单元。但用 cell 函数创建将会更加地高效，所以我们用 cell 创建大的单元数组。

7.2.1.1 用赋值语句创建单元阵列

你可以用赋值语句把值赋于单元阵列的一个单元，一次赋一个单元。这里有两种赋值的方法，即内容索引(content indexing)和单元索引(cell indexing)。

内容索引要用到大括号{}，还有它们的下标，以及单元的内容。例如下面的语句创建了一个 2×2 的单元阵列，如图 7.2 所示。

```
a{1,1} = [1 3 -7; 2 0 6; 0 5 1];
a{1,2} = 'This is a text string.';
a{2,1} = [3+4*i -5; -10*i 3-4*i];
```

```
a{2,2} = [];
```

索引的这种类型定义了包含在一个单元中的数据结构的内容。

单元索引把存储于单元中的数据用大括号括起来，单元的下标用普通下标标记法。例如下面的语句将创建一个 2×2 的单元阵列，如图 7.2 所示。

```
a(1,1) ={[1 3 -7; 2 0 6;0 5 1]};
a(1,2) = {'This is a text string.'};
a(2,1) = {[3+4*i -5; -10*i 3-4*i]};
a(2,2) = {[]};
```

索引的这种类型创建了包含有指定值的一个数据结构，并把这个数据结构赋于一个单元。

这两种形式是完全等价的，你可以在你的程序任选其一。

编程隐患

不要创建一个与已存在的数字数组重名的元阵列。如果得名了，MATLAB 会认为你把单元阵列的内容赋值给一个普通的数组，这将会产生一个错误信息。在创建单元阵列之前，确保同名的数字数字数组已经被删除。

7.2.1.2 用 cell 函数创建单元阵列

函数 cell 允许用户创建空单元阵列，并指定阵列的大小。例如，下面的语句创建一个 2×2 的空单元阵列。

```
a = cell(2, 2)
```

一旦单元阵列被创立，你就可以用赋值语句对单元阵列进行赋值。

7.2.2 单元创建者——大括号({})的应用

如果在单个大括号中列出所有单元的内容，那么就定义了许多的单元，在一行中的独立单元用逗号分开，行与行之间用分号隔开。例如下面的语句创建一个 2×3 单元阵列。

```
b = {[1 2], 17, [2;4]; 3-4*i, 'Hello', eye(3)}
```

7.2.3 查看单元阵列的内容

MATLAB 可以把单元阵列每一个元素的数据结构缩合在一行中显示出来。如果全部的数据结构没有被显示出来，那么显示就是一个总结。例如，单元阵列 a 和 b 显示如下

```
>> a
a =
    [3x3 double]    [1x22 char]
    [2x2 double]             []
>> b
b =
        [1x2 double]    [    17]    [2x1 double]
    [3.0000- 4.0000i]    'Hello'    [3x3 double]
```

注意 **MATLAB** 显示的只是数据结构，包括中括号和省略号，而不包含数据结构的内容。

如果你想要知道看到单元阵列的所有内容，要用到 celldisp 函数。这个函数显示的是每一个单元中的数据结构的内容。

```
>> celldisp(a)
a{1,1} =
        1        3       -7
        2        0        6
        0        5        1
a{2,1} =
     3.0000 + 4.0000i    -5.0000
             0 -10.0000i      3.0000 - 4.0000i
a{1,2} =
This is a text string.
a{2,2} =
        []
```

如果要用高质量的图象显示数据结构的内容，要用到函数 cellplot。例如，函数 cellplot(b) 产生了一个图象，如图 7.3 所示。

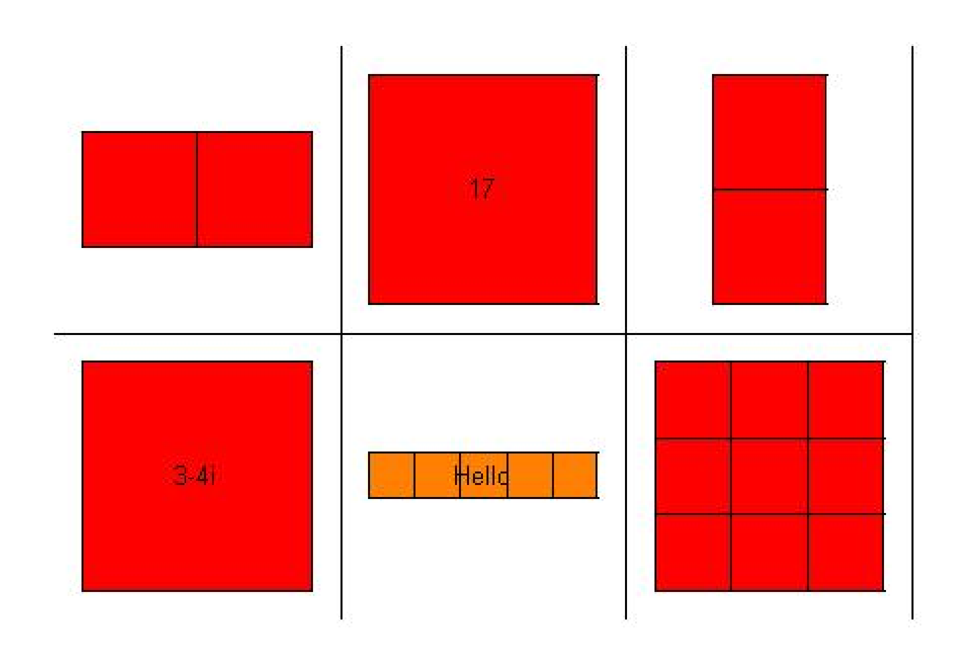


图 7.3 用函数 cellplot 显示单元阵列 b 数据结构的内容

7.2.4 对单元阵列进行扩展

一个值赋值于一个单元阵列中的元素，如果这个元素现在不存在，那么这个元素就会被自动的建立，其他所需的元素也会被自动建立。例如，假设定义了一个 2×2 单元阵列，如图 7.1 所示。如果我们执行下面的语句

a{3, 3} = 5

单元阵列将会自动扩展为 3×3 单元阵列，如图 7.4 所示。

<table>
<tr><td>cell 1,1
[1 3 -7; 2 0 5; 0 5 1]</td><td>cell 1,2
'This is a text string.'</td><td>cell 1,3
[]</td></tr>
<tr><td>cell 2,1
[3+i4 -5; -i10 3-i4]</td><td>cell 2,2
[]</td><td>cell 2,3
[]</td></tr>
<tr><td>cell 3,1
[]</td><td>cell 3,2
[]</td><td>cell 3,3
[5]</td></tr>
</table>

图 7.4 把一个值赋值于 a(3,3)产生的结果。注意其他的空元素也是自动创建的。

7.2.5 删除阵列中的元素

如果要删除阵列中的所有元素，我们要用 clear 命令。如果要删除单元阵列中的部分元素，我们把空值赋值于这一部分元素。例如，假设 a 的定义如下

```
>> a
a =
    [3x3 double]    [1x22 char]    []
    [2x2 double]             []    []
              []             []   [5]
```

我们可以用下面的语句删除第三行

```
>> a(3,:)=[]
a =
    [3x3 double]    [1x22 char]    []
    [2x2 double]             []    []
```

7.2.6 单元阵列数据的应用

在一个单元阵列中，数据结构中数据可以随时用内容索引或单元索引调用。

例如假设单元阵列 c 的定义如下

```
c = {[1 2; 3 4],'dogs';'cats',i}
```

存储于 c(1,1)的内容可由下面的语句调用

```
>> c{1,1}
ans =
     1     2
     3     4
```

同样 c(2,1)中的元素可由下面的元素调用

```
>> c{2,1}
ans =
cats
```

一个单元内容的子集可由两套下标得到。例如，假设我们要得到单元 c(1,1)中的元素(1,2)。为了达到此目的，我们可以用表达式 c{1,1}(1,2)，它代表单元 c(1,1)中的元素(1,2)。

```
>> c{1,1}(1,2)
ans =
```

```
2
```

7.2.7 字符串单元阵列

在一个单元阵列中存储一批字符串与在标准的字符数组中存储相比是非常方便的，因为在单元阵列中每一个字符串的长度可以是不相同的，而在标准字符数组的每一行的长度都必须相等。这就意味着在单元阵列中的字符串没的必要增加多余的空格。许多的 **MATLAB** 用户图形界面函数均使用单元阵列，正是基于这个原因，我们将在第十章看到。

字符串单元阵列可以由两种方法创建。我们可以用方括号把独立的字符串插入到单元阵列，我们也可以函数 cellstr 把一个二维字符数组转化为相应的字符串单元阵列。

下面的例子用第一种方法创建了一个字符串单元阵列，并显示出这个阵列的结果。注意下面的每一个字符串具有不同的长度。

```
>> cellstring{1} = 'Stephen J. Chapman';
>> cellstring{2} = 'Male';
>> cellstring{3} = 'SSN 999-99-9999';
>> cellstring
cellstring =
    'Stephen J. Chapman'    'Male'    'SSN 999-99-9999'
```

我们可以利用函数 cellstr 把一个二维字符数据转化为相应的字符串单元阵列。考虑下面的字符数组。

```
>> data = ['Line 1          ';'Additional Line']
data =
Line 1
Additional Line
```

相应的字符串单元阵列为

```
>> c = cellstr(data)
c =
    'Line 1'
    'Additional Line'
```

我们还可以用 char 函数它转化回去

```
>> newdata = char(c)
newdata =
Line 1
Additional Line
```

7.2.8 单元阵列的重要性

单元阵列是非常灵活的，因为各种类型的大量数据可以存储在每一个单元中。所以，它经常当作中间 **MATLAB** 数据结构来用。我们必须理解它，因为在第十章中 **MATLAB** 图形用户界面要用到它的许多特性。

还有，单元阵列的灵活性可能使它们具有函数普通特性，这个函数是指带有输入参数和输出参数的变量个数的函数。一种特别的输入参数 varargin 可以在自定义函数中得到，这种函数支持输入参数的变量的个数。这个参数显在输入参数列表的最后一项，它返回一个单元阵列，所以一个输入实参可以包括任意数目的实参。每一个实参都变成了由 varagin 返回的单元阵列元素。如果它被应用，varagin 必须是函数中的最后一个输入参数。

例如，假设我们要编写一个函数，它可能需要任意个数的输入参数。这个函数执行如下所示

```
function test1(varargin)
disp(['There are ' int2str(nargin) ' arguments.']);
disp('The input arguments are:');
disp(varargin);
```

我们用不同的数目参数来执行这个函数，结果如下

```
>> test1
There are 0 arguments.
The input arguments are:
>> test1(6)
There are 1 arguments.
The input arguments are:
    [6]
>> test1(1,'test 1',[1 2,3 4])
There are 3 arguments.
The input arguments are:
    [1]    'test 1'    [1x4 double]
```

正如我们所看到的，参数变成了函数中的单元阵列元素。

下面是一个简单函数例子，这个函数拥有不同的参数数目。函数 plotline 任意数目的 1×2 行向量，每一个向量包含一个点(x,y)。函数把这些点连成线。注意这个函数也接受直线类型字符串，并把这些字符串转递给 plot 的函数。

```
function plotline(varargin)
%PLOTLINE Plot points specified by [x,y] pairs.
% Function PLOTLINE accepts an arbitrary number of
% [x,y] points and plots a line connecting them.
% In addition, it can accept a line specification
% string, and pass that string on to function plot.
% Define variables:
% ii            --Index variable
% jj            --Index variable
% linespec      --String defining plot characteristics
% msg           --Error message
% varargin      --Cell array containing input arguments
% x             --x values to plot
% y             --y values to plot
% Record of revisions:
% Date      Programmer      Description of change
% ====      ==========      =====================
% 10/20/98 S. J. Chapman    Original code
% Check for a legal number of input arguments.
% We need at least 2 points to plot a line...
msg = nargchk(2,Inf,nargin);
error(msg);
% Initialize values
jj = 0;
linespec = '';
% Get the x and y values, making sure to save the line
% specification string, if one exists.
for ii = 1:nargin
    % Is this argument an [x,y] pair or the line
    % specification?
    if ischar(varargin{ii})
        % Save line specification
        linespec = varargin{ii};
    else
        % This is an [x,y] pair. Recover the values.
        jj = jj + 1;
```

```
            x(jj) = varargin{ii}(1);
            y(jj) = varargin{ii}(2);
        end
    end
    % Plot function.
    if isempty(linespec)
        plot(x,y);
    else
        plot(x,y,linespec);
    end
```

我们用下面的参数调用这个函数，产生的图象如图 7.5 所示。用相同的数目的参数调用函数，看它产生的结果为什么？

plotline([0 0], [1 1], [2 4], [3 9], 'k--');

也有专门的输出参数，vargout，它支持不同数目的输出参数。这个参数显示在输出参数列表的最后一项。它返回一个单无阵列，所示单个输出实参支持任意数目的实参。每一个实参都是这个单无阵列的元素，存储在 varargout。如果它被应用，varargout 必须是输出参数列表中最后一项，在其它输入参数之后。存储在 varargout 中的变量数由函数 nargout 确定，这个函数用指定于任何一个已知函数的输出实参。例如，我们要编写一函数，它返回任意数目的随机数。我们的函数可以用函数 nargout 指定输出函数的数目，并把这些数目存储在单元阵列 varargout 中。

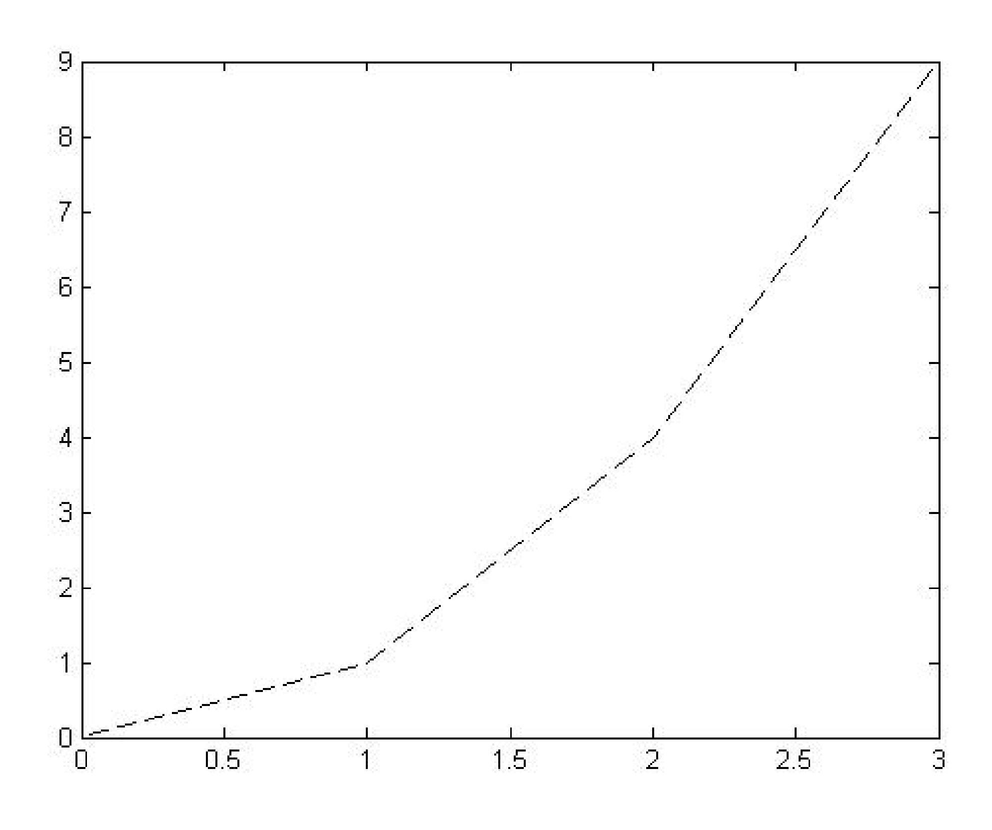


图 7.5 函数 plotline 产生的图象

```
function [nvals, varargout] = test2(mult)
%   nvals is the number of random values returned
%   varargout contains the random values returned
nvals = nargout - 1;
for ii = 1:nargout-1
    varargout{ii} =randn * mult;
end
```

当这个函数被执行时，产生的结果如下

```
>> test2(4)
ans =
     -1
>> [a b c d] = test2(4)
a =
      3
b =
   -1.7303
c =
   -6.6623
d =
     0.5013
```

好的编程习惯

应用单元阵列 varargin 和 varargout 创建函数，这个函数支持不同数目的输入或输出参数。

7.2.9 单元阵列函数总结

支持单元阵列的一些普通函数总结在表 7.2 中。

表 7.2 普通的单元阵列函数

函数	描述
cell	对单元阵列进行预定义
celldisp	显示出单元阵列的内容
cellplot	画出单元阵列的结构图
cellstr	把二维字符数组转化为相应的字符串单元阵列
char	把字符串单元阵列转化相应的字符数组

7.3 结构数组

一个数组是一个数据类型，这种数组类型有一个名字，但是在这个数组中的单个元素只能通过已知的数字进行访问。数组 arr 中的第五个元素可由 arr(5)访问。注意在这个数组中的所有元素都必须是同一类型(数字或字符)。一个单元阵列也是一种数据类型，也有一个名字，单个元素也只能通过已知的数字进行访问。但是这个单元阵列中元素的数据类型可以是不同的。相对地，一个**结构**也是一种数据类型，它的每一个元素都有一个名字。我们称结构中的元素为**域**。单个的域可以通过结构名和域名来访问，用句号隔开。

7.3.1 创建结构

创建结构有两种方法

- 用赋值语句创建
- 用函数 struct 函数进行创建

7.3.1.1 用赋值语句创建函数

你可以用赋值语句一次创建一个结构域。每一次把数据赋值于一个域，这个域就会被自动创建。例如用下面的语句创建如图 7.6 所示的结构。

```
>> student.name = 'John Doe';
>> student.addr1 = '123 Main Street';
>> student.city = 'Anytown';
>> student.zip = '71211'
student =
         name: 'John Doe'
        addr1: '123 Main Street'
         city: 'Anytown'
          zip: '71211'
```

第二个 student 可以通过在结构名前加上一个下标的方式加入到这个结构中。

```
>> student(2).name = 'Jane Q. Public'
student =
1x2 struct array with fields:
        name
        addr1
        city
        zip
```

Student 现在是一个 1×2 数据。注意当一个结构数据超一个元素，只有域名，而没有它的内容。在命令窗口中独立键入每一个元素，它的内容就会被列出。

```
>> student(1)
ans =
         name: 'John Doe'
        addr1: '123 Main Street'
         city: 'Anytown'
          zip: '71211'
>> student(2)
ans =
         name: 'Jane Q. Public'
        addr1: []
         city: []
          zip: []
```

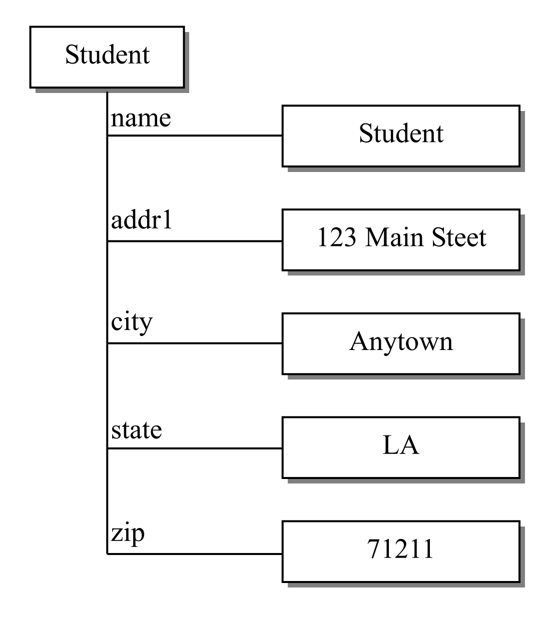


图 7.6 一个简单的例子。结构中每一个元素被称作域，每一个域都可以通过它的名字来访问。

注意无论结构的元素什么时间被定义，结构中的所有域都会被创建，不管它是否被初始化，没有被初始化的域将包含一个空数组，在后面我们可以用赋值语句来初始化这个域。

我们可以用 fieldnames 函数随时对结构中的域名进行恢复。这个函数可以返回一系列字

符单元阵列中的域名，这个函数对结构数组的运算是非常有用的。

7.3.1.2 用 struct 函数创建结构

函数 struct 允许用户预分配一个结构数据。它的基本形式如下

```
structure_array = struct(fields)
```

其中 fields 是填补结构域名的字符串数组或单元阵列。用上面的语法，函数 struct 用空矩阵初始化数组中的每一个域。

当域被定义时，我们就可以对它进行初始化，形式如下

```
str_array = struct('field1', var1, 'field2', val2, ...)
```

其中参数为域名和它们的值。

7.3.2 增加域到结构

如果一个新的域名在结构数组中的任意一个元素中被创建，那么这个域将会增加到数组的所有元素中去。例如，假设我们把一些成绩添加到 jane public 的记录中。

```
>> student(2).exams = [90 82 88]
student =
1x2 struct array with fields:
    name
    addr1
    city
    zip
    exams
```

如下所示，在数组的每一个记录中都有一个 exams 域。这个将会在 student(2)中进行初始化，其他同学的数组为空，除非用赋值语句给他赋值。

```
>> student(1)
ans =
      name: 'John Doe'
     addr1: '123 Main Street'
      city: 'Anytown'
       zip: '71211'
     exams: []
>> student(2)
ans =
      name: 'Jane Q. Public'
     addr1: []
      city: []
       zip: []
     exams: [90 82 88]
```

7.3.3 删除结构中的域

我们可以用 rmfield 函数删除结构数据中的域。这个函数的形式如下

```
struct2 = rmfield(struct_array, 'field')
```

其中 struct_array 是一个结构数组，field 是要去除的域，stuct2 是得到的新结构的名字。例如从结构 student 中去除域名 zip，代码如下

```
>> stu2 = rmfield(student, 'zip')
```

```
stu2 =

1x2 struct array with fields:
       name
       addr1
       city
       exams
```

7.3.4 结构数组中数组的应用

现在我们假设结构 sdudent 中已经有三个学生，所有数据如图 7.7 所示。我们如何应用结构数组中的数据呢？

我们可以在句号和域名后加数组元素名访问任意数组元素的信息。

```
>> student(2).addr1
ans =
P.O.Box 17
>> student(3).exams
ans =
       65       84       81
```

任何带一个域的独立条目可以在域名后加下标的方式进行访问，例如，第三个学生的第二个科目的成绩为

```
>> student(3).exams(2)
ans =
       84
```

结构数组中的域能作为支持这种数组类型任意函数的参数。例如，计算 student(2)的数据平均值，我们使用这个函数

```
>> mean(student(2).exams)
ans =
     86.6667
```

不幸是的，我们不能同时从多个元素中得到一个域的值。例如，语句

student.name 是无意义的，并导致错误。如果我们想得到所有学生的名字，我们必须用到一个 for 循环。

```
for ii = 1:length(student)
       disp(student(ii).name);
end
```

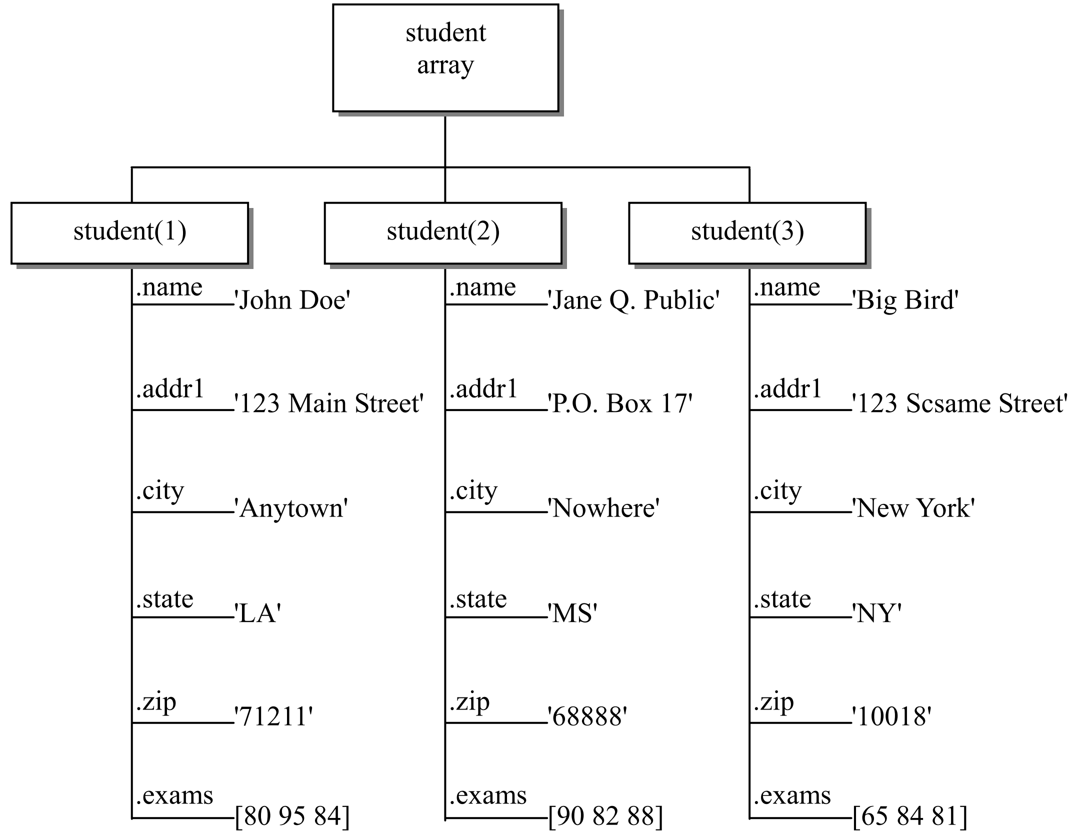


图 7.7 用三个元素组成的 student 数组和它所有的域

相似地，如果我们想得到所有学生的平均成绩，我们不能用函数 mean(student.exams)，我们必须独立的访问学生的每一个成绩，并算出总成绩。

```
exam_list = [];
for ii = 1:length(student)
    exam_list = [exam_list student(ii).exams];
end
mean(exam_list)
```

7.3.5 函数 getfield 和函数 setfield

MATLAB 中有两个函数用来创建结构，这比用自定义函数创建要简单的多。函数 getfield 得到当前存储在域中的值，函数 setfield 在域中插入一新值。

getfield 函数的结构是

```
f = getfield(array,{array_index},'field',{field_index})
```

field_index 和 array_index 是可选择性，array_index 用于创建 1×1 结构数组。调用这个函数的语句为

```
f = array(array_index).field(field_index);
```

当编写程序时，尽管程序不知道结构数组中的域名，这个语句也可以应用。例如，假设我们需要编写一个程序，读取一未知数组并对它进行操作。函数可以通过调用 fieldnames 命令来确定一个结构数据的域名，然后通过函数 getfield 读取数据。为了读取第二个学生的 zip 码，函数为

```
>> zip = getfield(student,{2},'zip')
zip =
68888
```

相似地，我们可以用 setfield 函数修改结构的值。形式如下

```
f = setfield(array,{array_index},'field',{field_index},value)
```

f 是输出结构数组，field_index 和 array_index 都是可选择性参数，array_index 用于创建 1×1 结构数组。调用这个函数的语句为

```
array(array_index).field(field_index) = value;
```

7.3.6 对结构数组应用 size 函数

当 size 函数应用于结构数组时，它将返回这个结构数组的大小。当 size 函数应用于结构数组中的一个元素的域时它将返回这个域的大小。例如

```
>> size(student)
ans =
        1        3
>> size(student(1).name)
ans =
        1        8
```

7.3.7 结构的嵌套

结构数组的每一个域可是任意数据类型，包括单元阵列或结构数组。例如，下面的语句定义了一个新结构数组。它作为 student 的一个域，用来存储每一个学生所在班级的信息。

```
>> student(1).class(1).name = 'COSC 2021';
>> student(1).class(2).name = 'PHYS 1001';
>> student(1).class(1).instructor = 'Mr. Jones';
>> student(1).class(2).instructor = 'Mrs. Smith';
```

在这个语句被执行后，student(1)将由以下数据组成。注意访问嵌套结构中的数据方法。

```
>> student(1)
ans =
      name: 'John Doe'
     addr1: '123 Main Street'
      city: 'Anytown'
     state: 'LA'
       zip: '71211'
     exams: [80 95 84]
     class: [1x2 struct]
>> student(1).class
ans =
1x2 struct array with fields:
    name
    instructor
>> student(1).class(1)
ans =
          name: 'COSC 2021'
    instructor: 'Mr. Jones'
>> student(1).class(2)
ans =
          name: 'PHYS 1001'
    instructor: 'Mrs. Smith'
>> student(1).class(2).name
ans =
```

7.3.8 struct 函数总结

支持 struct 的普通函数总结在表 7.3 中。

表 7.3 支持 struct 的函数

函数	描述
fieldnames	在一个字符串单元阵列中返回域名
getfield	从一个域中得到当前的值
rmfield	从一个结构中删除一个域
setfield	在一个域中设置一个新值
struct	预定义一个结构

测试 7.1

本测试提供了一个快速的检查方式，看你是否掌握了本章的基本内容。如果你对本测试有疑问，你可以重读本章，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. 什么是稀疏矩阵？它和全矩阵有何区别？两者之间如何相互转化？
2. 什么是单元阵列？它和普通的数组有何区别？
3. 内容检索和单元检索有何区别？
4. 什么是结构？它与稀疏矩阵，单元阵列有什么区别？
5. varargin 的功能是什么？它是如何工作的？
6. 下面给出了数组 a 的定义，下面的语句将会产生怎样的结果？

```
a{1,1} = [1 2 3; 4 5 6; 7 8 9];
a(1,2) = {'Comment line'};
a{2,1} = j;
a{2,2} = a{1,1} -a{1,1}(2,2);
```

(a) a(1,1)　(b) a{1,1}　(c) 2*a(1,1)　(d) 2*a{1,1}
(e) a{2,2}　(f) a(2,3) = {[-17;17]}　(g) a{2,2}(2,2)

7. 下面给出了结构 b 的定义，下面的语句将会产生怎样的结果？

```
b(1).a = -2*eye(3);
b(1).b = 'Element 1';
b(1).c = [1 2 3];
b(2).a = (b(1).c' [-1; -2; -3] b(1).c');
b(2).b = 'Element 2';
b(2).c = [1 0 -1];
```

(a) b(1).a - b(2).a　(b) strncmp(b(1).b, b(2).b,6)
(c) mean(b(1).c)　(d) mean(b.c)
(e) b　(f) b(1)

7.4 总结

本章重点介绍了三类数据类型：稀疏矩阵，单元阵列和结构。

7.4.1 好的编程习惯总结

当你访问一单元阵列时，不要把()与{}混淆。它们完全不同的运算。

7.4.2 MATLAB 函数命令总结

普通的 **MATLAB** 稀疏矩阵函数

类别	函数	描述
创建一个稀疏矩阵	speye	创建一个单位稀疏矩阵
	sprand	创建一个稀疏矩阵，元素是符合平均分布的随机数
	sprandn	创建一个稀疏矩阵，元素是普通的随机数
全矩阵和稀疏矩阵的转换函数	sparse	把一个全矩阵转化为一个稀疏矩阵
	full	把一个稀疏矩阵转化为全矩阵
	find	找出矩阵中非零元素和它对应的上下标
对稀疏矩阵进行操作的函数	nnz	非零元素的个数
	nonzeros	返回一个向量，其中的元素为矩阵中非零元素
	spones	用 1 代替矩阵中的非零元素
	spalloc	一个稀疏矩阵所占的内存空间
	issparse	如果是稀疏矩阵就返回 1
	spfun	给矩阵中的非零元素提供函数
	spy	用图象显示稀疏矩阵

普通的单元阵列函数

函数	描述
cell	对单元阵列进行预定义
celldisp	显示出单元阵列的内容
cellplot	画出单元阵列的结构图
cellstr	把二维字符数组转化为相应的字符串单元阵列
char	把字符串单元阵列转化相应的字符数组

支持 struct 的函数

函数	描述
fieldnames	在一个字符串单元阵列中返回域名
getfield	从一个域中得到当前的值
rmfield	从一个结构中删除一个域
setfield	在一个域中设置一个新值
struct	预定义一个结构

7.5 练习

7.1 编写一个 **MATLAB** 函数，可以接爱一个字符串单元阵列，并根据 ascii 码字母顺序对它进行升序排列。(如果你愿意的话，可以利用第六章的函数 c_strcmp 对它们进行比较。)

7.2 编写一个 **MATLAB** 函数，接受一个字符串单元阵列，并按字母表的顺序进行排序。(注意在这里不区分大小写)

7.3 创建一个 100×100 的稀疏矩阵，其中 5%的元素是按普通分布的随机数(用 sprandn 产生这些值)，其余为 0。下一步，把数组对角线上的所有元素都设置为 1。下一步，定义一个含 100 个元素稀疏列向量 b，并用 100 个符合平均分布的随机数赋值于 b。回答下面的问题。

a. 利用稀疏矩阵 a 创建一个全矩阵 a_full。比较两矩阵所需的内存？那一个更高效呢？

b. 应用 spy 函数画出 a 中元素的分布

c. 利用稀疏矩阵 b 创建一个全矩阵 b_full。比较两矩阵所需的内存？那一个更高效呢？

d. 分别用全矩阵和稀疏矩阵角方程组 a*x=b 中未知矩阵 x 的值？

7.4 创建一个函数，接受任意数目数字输入参数，计算出所有参数中单个元素的总和。用下面 4 个参数检测你的程序

$a = 10$， $b = \begin{bmatrix} 4 \\ -2 \\ 2 \end{bmatrix}$， $c = \begin{bmatrix} 1 & 0 & 3 \\ -5 & 1 & 2 \\ 1 & 2 & 0 \end{bmatrix}$， d = [1 5 -2]。

7.5 修改先前练习中的函数，使它既能够接受数字数组又能接受包含数字值的单元阵列。

用下面 2 组参数检测你的程序。

$a = \begin{bmatrix} 1 & 4 \\ -2 & 3 \end{bmatrix}$， b{1} = [1 5 2]，和 $b\{2\} = \begin{bmatrix} 1 & -2 \\ 2 & 1 \end{bmatrix}$

7.6 创建一个结构，画图的所有信息，在取最小值时，结构数据应当有下面的域

x_data	x 的值
y_data	y 的值
type	线性，对数等
plot_title	图象标题
x_label	x 轴标签

你也可以增加其他的域来增强你对最终图象的控制。

在结构数据被创建后，创建一个函数，它能接受包含有这个结构的数组，在这个数组中每一个结构产生一个图象。这个函数应当支持一些默认的域值。例如，如果域 plot_title 是一个空矩阵，那么产生的图象将无标题材。在编写程序之前，想好你的默认值。

为了检测你的函数，创建一个结构，包括创建三个不同图形的数据，并把这个结构传递给你的函数。这个函数应当正确的画出三个图形。

第八章 输入/输出函数

在第二章中，我们已经学到如何用 load 和 save 命令加载和保存 **MATLAB** 数据，以及如何使用 fprintf 函数格式化输出数据。在本章中，我们将学习更多的关于 **MATLAB** 输入和输出的功能。首先，我们将会学习函数 textread，在 maltab5。3 中它是一个非常有用的函数。然后，我们将花更多的时间学习 load 和 save 命令。最后，我们将查看其他的 **MATLAB**I/O 选择。

熟悉 C 语言的读者对这部分数据将会十分的熟悉。但是，在 **MATLAB** 函数和 c 函数之间有细微的不同。

8.1 函数 textread

命令 textread 最早出现于 **MATLAB**5.3 中。它可以按列读取 ascii 文件中的元素，每一列中可能含有不同的数据类型。这函数读取其他程序生成的数据表时非常地有用。

这个命令的形式如下

```
[a, b, c, ...] = textread(filename, format, n)
```

其中 filename 代表要打开的文件的名字，format 是用于每一行数据类型的字符串，n 代表要读取的行数(如果没有 n，则这个命令将读完这个文件)。格式化字符串与函数 fprintf 格式化描述的字符串相同。注意输出参数的个数必须与你读取的列数相区配。

例如，假设文件 test_input.dat 包含下列数据

```
James Jones O+ 3.51 22 Yes
Sally Smith A+ 3.28 23 NO
```

这些数据用下面的函数读取一系列的数组。

```
[first, last, blood, gpa, age, answer] = textread('test_input.dat','%s %s %s %f %d %s')
```

当这个函数被编译时产生如下结果

```
>> [first, last, blood, gpa, age, answer] = textread('test_input.dat','%s %s %s %f %d
%s')
first =
    'James'
    'Sally'
last =
    'Jones'
    'Smith'
blood =
    'O+'
    'A+'
gpa =
    3.5100
    3.2800
age =
    22
    23
answer =
    'Yes'
    'NO'
```

这个函数可以通过在格式描述符前面加一个星号的方式来跳过某些所选项。例如，下面

的语句只从文件只读取 first，last 和 gpa。

```
>> [first, last, gpa] = textread('test_input.dat','%s %s %*s %f %*d %*s')
first =
       'James'
       'Sally'
last =
       'Jones'
       'Smith'
gpa =
       3.5100
       3.2800
```

函数 textread 要比 load 命令简单有效的多。load 命令假设输入文件中的所有数据都是同一类型——它不支持在不同的列上有不同的数据。此外，它把所有的数据都存储在一个数据中。相反地，函数 textread 允许每一列都有独立的变量，当和由不同类型的数据组成的列运算时，它更加的方便。

函数 textread 中有许许多多参数，它们增加了函数的灵活性。你可通过咨询 **MATLAB** 的在线文本得到这些参数的使用细节。

编程隐患

应用函数 text 从 ascii 文件中按行格式读取数据，这个 ascii 文件可能是其他语言生成的，或是由其他的应用程序生成的，例如表格。

8.2 关于 load 和 save 命令的进一步说明

save 命令把 **MATLAB** 工作区数据存储到硬盘，load 命令把硬盘上的数据拷贝到工作区中。save 命令即可用特殊的二进制格式 mat- file 存储数据，也可用普通的 ascii 码格式存储数据。save 命令的形式为

```
save filename [list of variables] [options]
```

如果只有 save 命令，那么当前工作区内的所有数据存储在一个名为 **MATLAB**.mat 的文件中。如果后面有一个文件名，那么这些数据将会存储在“filename.mat”的文件。如果后面还包括一系列的变量，那么就只存储这些特殊的变量。

支持 save 命令的参数如表 8.1 所示。

表 8.1save 命令的参数

参数	描述
- mat	以 mat 文件格式存储数据（默认）
- ascii	用 ascii 格式保存数据
- append	给已存在 matf 文件增加变量
- v4	也存储为 mat 文件格式，但能被 **MATLAB**4.0 读取

load 命令可以加载 mat 文件或普通的 ascii 文件中的数据。load 命令的形式如下

```
load filename [option]
```

如果只有 load 命令，**MATLAB** 将加载 **MATLAB**.mat 文件中的所有数据。如果还跟着一个文件名，它 load 命令将会加载这个文件中的数据。

支持 load 命令的参数被列于表 8.1 中。

尽管它们的优点不是十分的明显，但是 save 和 load 命令是 **MATLAB** 中功能最强大，最有用的 I/O 命令。它的优点是

1. 这些命令易于使用
2. mat 文件的平台独立。在一个支持 **MATLAB** 的计算机上编写的文件，在另一种支持 **MATLAB** 的计算机上，可以被读取。这种格式可以在 PC，Mac，许多不同版本的 Unix

上互相转换。

3. mat 文件高效的硬盘空间使用者，它存储数据是高精度的，在 mat 文件和 ascii 文件转化过程中会出现精度下降的情况。

4. mat 文件存储了工作区内的每一个变量的所有信息，包括它的类属，名字和它是不是全局变量。在 I/O 其他类型数据存储格式中所有的这些信息都会丢失。例如，假设工作区包含下面信息。

```
>> whos
  Name          Size                Bytes   Class
  a             10x10                800    double array (global)
  ans           1x1                    8    double array
  b             10x10                800    double array
  c             2x2                  332    cell array
  string        1x16                  32    char array
  student       1x3                 2152    struct array
Grand total is 372 elements using 4124 bytes
```

如果工作区用 save workspace.mat 命令存储，那么文件 workspace.mat 就会被自动创建。当这个文件被加载时，工作区中的所有信息都会被恢复，包括每一项的类型和一变量是否为全局变量。

这个命令的缺点是生成的 mat 文件只能由 **MATLAB** 调用，其他的程序不可能利用他共享数据。如要你想要与其他程序共享数据，可以应用-ascii 参数，但它有诸多的限制。

表 8.2 load 命令参数

参数	描述
-mat	把文件当作 mat 文件看待（如果扩展名是 mat，此为默认格式）
-ascii	把文件当作 ascii 格式文件来看待（如果扩展名不为 mat，此为默认格式）

好的编程习惯

除非我们必须与非 MATLAB 程序进行数据交换，存储和加载文件时，都应用 mat 文件格式。这种格式是高效的且移植性强，它保存了所有 MATLAB 数据类型的细节。

save -ascii 根本不能存储单元阵列和结构数据，在保存字符串之前，它要把字符串转化相应的数字形式。load -ascii 命令只能加载空间独立的数据，这些数据每一行的元素个数都相等，**MATLAB** 把所有的数据都存储于一个变量中，这个变量与输出文件同名。如果你要用更高的要求（例如，保存和加载字符串，单元阵列或结构数组并与其它程序进行交换），那么你需要本章后面介绍的 I/O 命令。

如果我们要加载的文件名或变量名是字符串，那么我们要用这些命令的函数形式。例如，下面的代码段要求用户提供一个文件名，并把当前工作区保存在那个文件中。

```
filename = input('Enter save file name: ','s');
save(filename);
```

8.3 MATLAB 文件过程简介

为了使用在 **MATLAB** 程序中的文件我们需要一些方法选出我们所要的文件，并从中读取或写入数据。在 **MATLAB** 中有一种非常灵活的读取/写入文件的方法，不管这个文件是在磁盘还是在磁带上或者是其他的存储介质。这种机制就叫做文件标识（file id）（有时可简写为 fid），当文件被打开，读取，写入或操作时，文件标识是赋值于一个文件的数。文件标识是一个正整数。两种文件标识是公开的——文件标识 1 是标准输出机制，文件标识 2 是标准错误机制（stderr）。其他的文件标识，在文件打开时创立，文件关闭时消逝。

许多的 **MATLAB** 语句可以控制磁盘文件的输入或输出。文件 I/O 函数被总结在表 8.3 中。

表 8.3 **MATLAB** 输入/输出语句

类别	函数	描述
加载/保存工作区	load	加载工作区
	save	保存工作区
文件打开/关闭	fopen	打开文件
	fclose	关闭文件
二进制 I/O	fread	从文件中读取二进制数据
	fwrite	把二进制数据写入文件
格式化 I/O	fscanf	从文件中读取格式化数据
	fprintf	把格式化数据写入文件
	fgetl	读取文件的一行，忽略换行符
	fgets	读取文件的一行，不忽略换行符
文件位置、状态	delete	删除文件
	exist	检查文件是否存在
	ferror	所需文件的 I/O 错误情况
	feof	检测文件的结尾
	fseek	设置文件的位置
	ftell	检查文件的位置
	frewind	回溯文件
临时情况	tempdir	得到临时目录名
	tempname	得到临时文件名

我们可以用 fopen 语句把文件标识传递给磁盘文件或设备，用 fclose 语句把他们从中分开。一旦一个文件用 fopen 语句得到一个文件标识，我们就可以利用 **MATLAB** 输入输出语句。当我们对这个文件操作完后，fclose 语句关闭并使文件标识无效。当文件打开时，函数 frewind 和 fseek 常用于改变当前文件读取和写入的位置。

在文件中读取或写入数据的方法有两种方法：像二进制数据或像格式化字符数据。由实际位样式组成的二进制数据常用于存储于计算机内存中。读取和编写二进制数据是非常高效的，但是用户不能读取存在于文件中的数据。在格式化文件中的可以转化为符串的数据可以由用户直接读取。格式化 I/O 操作比二进制 I/O 操作要慢得多，效率要低得多。在本章中，我们将讨论两种类型的 I/O 的操作。

8.4 文件的打开与关闭

文件的打开与关闭函数，fopen 和 fclose 将在本节描述。

8.4.1 fopen 函数

fopen 函数打开一个文件并对返回这个文件的文件标识数。它的基本形式如下：

```
fid = fopen(filename, permission)
[fid, message] = fopen(filename, permission)
[fid, message] = fopen(filename,permission, format)
```

其中 filename 是要打开的文件的名字，premission 用于指定打开文件的模式，format 是一个参数字符串，用于指定文件中数据的数字格式。如果文件被成功打开，在这个语句执行之后，fid 将为一个正整数，message 将为一个空字符。如果文件打开失败，在这个语句执行之后，fid 将为-1，message 将为解释错误出现的字符串。如果 **MATLAB** 要打开一个不为当前目录的文件，那么 **MATLAB** 将按 **MATLAB** 搜索路径搜索。

permisson 的字符串被列在表 8.4 中。

对于一些如 PC 一样的平台，它更重要的是区分文本文件和二进制文件。如果文件以文本格式打开，那么一个“t”就应加入到 permission 字符串中（例如“rt”或“rt+”）。

表 8.4 fopen 文件 permissions

文件 permission	意义
'r'	以只读格式读取文件
'r+'	可对文件进行读写
'w'	删除一个已存在文件的内容（或创建一个新文件），并以只写格式打开
'w+'	删除一个已存在文件的内容（或创建一个新文件），并以读写格式打开
'a'	打开一个已存在的文件（或创建一个新文件），并以只写文件格式打开把写入的内容增加到文件的结尾
'a+'	打开一个已存在的文件（或创建一个新文件），并以只写文件格式打开把写入的内容增加到文件的结尾
'W'	不进行自动洗带的写入数据（针对于磁带机的特殊命令）
'A'	不进行自动洗带的添加数据（针对于磁带机的特殊命令）

如果是以二进制模式找开，那么“b”应加到 permission 字符串中（例如“rb”）。这实际上是不需要的，因为文件默认打开的方式是二进制模式。文本文件和二进制文件在 Unix 系统上是没有区别的，所以在这系统上，r 和 b 都不需要。

在 fopen 函数中的 format 字符串数据存储在文件中的格式。当在两计算机中传递互相矛盾的数据格式时，这个字符串才是必须的。一些可能的数字格式被总结在表 8.5 中。你可以从 **MATLAB** reference manual（参考手册）中得到所有可能的数字格式。

这个函数有两种提供信息的格式。函数

```
fids = fopen('all')
```

返回一个行向量，这个行向量由当打开的所有文件的文件标识组成（除了 stdcut 和 stderr）。在这个向量中的元素的个数与所要打开的文件的个数相等。函数

```
[filename, permission, format] = fopen(fid)
```

对于一指定文件标识的打开文件，返回它的名字，权限（permission）字符串和数字格式。

下面是一些正确应用 fopen 函数的例子。

8.4.1.1 情况 1：为输入而打开一二进制文件

下面的函数只为输入二进制数据而打开文件 example.dat。

```
fid = fopen('example.dat','r')
```

权限（permission）字符串是“r”，它指出这个文件的打开方式为只读。这个字符串也可以是“rb”，但这是没有必要的，因为 **MATLAB** 默认打开的是二进制文件。

8.4.1.2 情况 2：为文本输出打开一文件

下面的函数以文本输出打开文件 outdat。

```
fid = fopen('outdat','wr')
```

或

```
fid = fopen('outdat','at')
```

权限字符串“wt”指定这个文件为新建文本文件。如果这个文件已存在，那旧文件就会被删除，打开新建的文件等待写入数据。如果我们要替换先前已存在的数据，那么就可以采用这个形式。

权限运算符“at”指定一个我们想要增加数据的文本文件。如果这个文件已经存在了，那么它将会被打开，新的数据将会添加到已存在的数据中。如果我们不想替换已存在的数据，那么就可以采用这个方式。

8.4.1.3 以读写模式打开文件

下面的函数打开文件 junk，可以对它进行二进制输入和输出。

```
fid = fopen('junk', 'r+')
```

或

```
fid = fopen('junk','w+')
```

每一个语句与第二个语句的不同为第一句打开已存在文件，而第二个语句则会删除已存在的文件。

好的编程习惯

在使用 fopen 语句时，一定要注意指定合适的权限，这取决于你是要读取数据，还是要写入数据。好的编程习惯可以帮助你避免（类似于覆盖的）错误。

在试图打开一个文件之后，检查错误是非常重要的。如果 fid 的值为-1，那么说明文件打开失败。你将把这个问题报告给用户，允许他们选择其他的文件或跳出程序。

好的编程习惯

在文件打开操作后检查它的状态以确保它被成功打开。如果文件打开失败，提示用户解决方法。

8.4.2 fclose 函数

fclose 函数用于关闭一文件。它的形式为

```
status = fclose(fid)
status = fclose('all')
```

其中 fid 为文件标识，status 是操作结果，如果操作成功，status 为 0，如果操作失败，status 为-1。

函数 status = fclose('all')关闭了所有的文件，除了 stdout（fid = 1）和 stderr（fid = 0）。如果所有的文件关闭成功，status 将为 0，否则为-1。

8.5 二进制 I/O 函数

二进制 I/O 函数，fwrite 和 fread，将在本节讨论。

8.5.1 fwrite 函数

函数 fwrite 以自定义格式把二进制数据写入一文件。它的形式为

```
count = fwrite(fid, array, precision)
count = fwrite(fid, array, precision skip)
```

其中 fid 是用于 fopen 打开的一个文件的文件标识，array 是写出变量的数组，count 是

写入文件变量的数目。

MATLAB 以列顺序输出数据，它的含义为第一列全部输出后，再输出第二列等等。例如，如果 $array = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$. 那么数据输出的顺序为 1，3，5，2，4，6。

参数 precision 字符串用于指定输出数据的格式。**MATLAB** 既支持平台独立的精度字符串，在所有的有 **MATLAB** 运行的电脑，它是相同的，也支持平台不独立的精度字符串，它们在不同类型的电脑上精度也不同。你应当只用平台独立的字符串，在本书中出现的字符串均为这种形式。

表 8.6 **MATLAB** 精度字符串

精度字符串	C/Fortan 形式	意义
'char'	'char*1'	6 位字符
'schar'	'signed char'	8 位有符号字符
'uchar'	'unsigned char'	8 位无符号字符`
'int8'	'integer*1'	8 位整数
'int16'	'integer*2'	16 位整数
'int32'	'integer*4'	32 位整数
'int64'	'integer*8'	64 位整数
'uint8'	'integer*1'	8 位无符号整数
'uint16'	'integer*2'	16 位无符号整数
'uint32'	'integer*4'	32 位无符号整数
'uint64'	'integer*8'	64 位无符号整数
'float32'	'real*4'	32 位浮点数
'float64'	'real*8'	64 位浮点数
'bitN'		N 位带符号整数(1≤N≤64)
'ubitN'		N 位无符号整数(1≤N≤64)

平台独立的精度显示在表 8.6 中。所有的这些精度都以字节为单位，除了“bitN”和“ubitN”，它以位为单位。

选择性参数 skip 指定在每一次写入输出文件之前要跳过的字节数。在替换有固定长度的值的时侯，这个参数将非常的有用。注意如果 precision 是一个像“bitN”或“ubitN”的一位格式，skip 则用位当作单位。

8.5.2 fread 函数

函数 fread 读取用用户自定义格式从一文件中读取二进制数据。它的格式如下

```
[array, count] = fread(fid, size, precision)
[array, count] = fread(fid, size, precision, skip)
```

其中 fid 是用于 fopen 打开的一个文件的文件标识，array 是包含有数据的数组，count 是读取文件中变量的数目，size 是要读取文件中变量的数目。

参数 size 用于指定读取文件中变量的数目。这个参数有三种形式。

- n　　准确地读取 n 个值。执行完相应的语句后，array 将是一个包含有 n 个值的列向量
- Inf　　读取文件中所有值。执行完相应的语句后，array 将是一个列向量，包含有从文件所有值。
- [n，m]　从文件中精确定地读取 n×m 个值。array 是一个 n×m 的数组。

如果 fread 到达文件的结尾，而输入流没有足够的位数写满指定精度的数组元素，fread 就会用最后一位的数填充，或用 0 填充，直到得到全部的值。如果发生了错误，读取将直接到达最后一位。

参数 precision 和 size 在函数 fread 和函数 fwrite 中有相同的意义。

例 8.1 读写二进制数据

在本例中显示的脚本文件创建了一个含有 10000 个随机数的数组，以只写方式打开一个自定义文件，用 64 位浮点数格式把这个数据写入磁盘，并关闭文件。程序打开所要读取的文件，并读取数组，得到一个 100×100 的数组。它用来说明二进制 I/O 操作。

```
% Script file: binary_io.m
%
% Purpose:
% To illustrate the use of binary i/o functions.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/19/98 S. J. Chapman Original code
%
% Define variables:
% count          -- Number of values read / written
% fid            -- File id
% filename       -- File name
% in_array       -- Input array
% msg            -- Open error message
% out_array      -- Output array
% status         -- Operation status
% Prompt for file name
filename = input('Enter file name: ','s');
% Generate the data array
out_array = randn(1,10000);
% Open the output file for writing.
[fid,msg] = fopen(filename,'w');
% Was the open successful?
if fid > 0
    % Write the output data.
    count = fwrite(fid,out_array,'float64');
    % Tell user
    disp([int2str(count) ' values written...']);
    % Close the file
    status = fclose(fid);
else
    % Output file open failed. Display message.
    disp(msg);
end
% Now try to recover the data. Open the
% file for reading.
[fid,msg] = fopen(filename,'r');
% Was the open successful?
if fid > 0
    % Read the input data.
    [in_array, count] = fread(fid,[100 100],'float64');
    % Tell user
    disp([int2str(count) ' values read...']);
    % Close the file
    status = fclose(fid);
```

```
else
	% Input file open failed. Display message.
	disp(msg);
end
```

当这个程序运行时，结果如下

```
>> binary_io
Enter file name: testfile
10000 values written...
10000 values read...
```

在当前目录下，有一个 80000 字节的文件 testfile 被创建，这个文件之所以占 80000 个字节，是因为它含有 10000 个 64 位的值，每一个值占 8 个字节。

测试 8.1

本测试提供了一个快速的检查方式，看你是否掌握了 8.1 到 8.5 的基本内容。如果你对本测试有疑问，你可以重读 8.1 到 8.5，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. 为什么函数 textread 尤其用于读取由其他的语言编写的数据。
2. 用 mat 存储数据的优缺点？
3. 哪些 **MATLAB** 函数用于找开和关闭文件？打开一个二进制文件和打开一个文本文件有什么不同？
4. 编写 **MATLAB** 语言找开一个已存在的文件 myinput.dat，把新增加的内容添加到最后一行。
5. 编写 **MATLAB** 语句，需要以只读模式打开一个无格式的二进制输入文件 input.dat。检测文件是否存在，如果不存在，它就产生一个合适的错误信息。

看第 6 题和第 7 题，判断 **MATLAB** 语句是否正确。如果有错误，指出错在那里。

6.
```
fid = fopen('file1', 'rt');
array = fread(fid, Inf)
fclose(fid);
```
7.
```
fid = fopen('file1', 'w');
x = 1:10;
count = fwrite(fid, x);
fclose(fid);
fid = fopen('file1','r');
array = fread(fid, [2 Inf])
fclose(fid);
```

8.6 格式化 I/O 函数

在本节中，我们向大家介绍格式化 I/O 函数。

8.6.1 fprint 函数

函数 fprint 把以户自定义格式编写的格式化数据写入一个文件。它的形式为

```
count = fprintf(fid, format, val1, val2, ...)
fprintf(format, val1, val2, ...)
```

其中 fid 是我们要写入数据那个文件的文件标识，format 是控制数据显示的字符串。如果 fid 丢失，数据将写入到标准输出设备（命令窗口）。这些格式已经在第二章介绍过。

格式（format）字符串指定队列长度，小数精度，域宽和输出格式的其他方面。它包括文字数字字符（%）和字符序列（用于指定输出数据显示的精确格式）。一个典型的数据输出格式字符串图 8.1 所示。字符%总是标志着格式化字符串的开始，在字符%之后，这字符串应包括一个标识（flag），一个域宽，一个精度指定符和一个转换指定符。字符%，转换指定符一般会要求出在任何格式中，而标识，域宽，精度指定符是可选的。

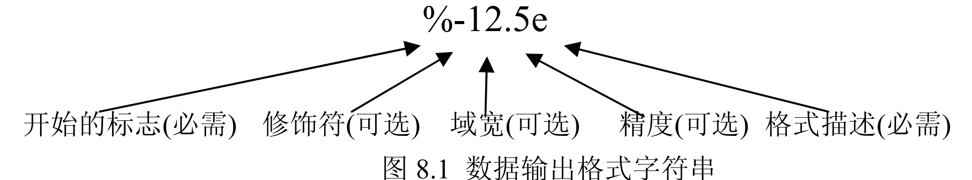


图 8.1 数据输出格式字符串

可能的转换指定符被列在表 8.7 中，可能的修改符（标识）被列在了表 8.8 中。如果我们用格式化字符串指定域宽和精度，那么小数点前的数就是域宽，域宽是所要显示的数所占的字符数。小数点后的数是精度，是指小数点后应保留的位数。

除了普通的字符和格式字符，还有转义字符常用在格式化字符串。这些特殊的字符被列在了表 8.9 中。

表 8.7 函数 fprintf 的格式转换指定符

指定符	描述
%c	单个字符
%d	十进制表示（有符号的）
%e	科学记数法（用到小写的 e，例 3.1416e+00）
%E	科学记数法（用到大写的 e，例 3.1416E+00）
%f	固定点显示
%g	%e 和%f 中的复杂形式，多余的零将会被舍去
%G	与%g 类似，只不过要用到大写的 E
%o	八进制表示（无符号的）
%s	字符串
%u	十进制（无符号的）
%h	用十六进制表示（用小写字母 a-f 表示）
%H	用十六进制表示（用大写字母 A-F 表示）

表 8.8 格式标识（修改符）

标识（修改符）	描述
负号(-)	数据在域中左对齐，如果没有这个符号默认为右对齐
+	输出时数据带有正负号
0	如果数据的位数不够，用零填充前面的数

表 8.9 格式字符串的转义字符

转义序列	描述
\n	换行
\t	水平制表
\b	退后一格
\r	回车符，使屏幕光标移到当前行开关，下移到下一行
\f	跳页符号
\\	打印一个普通反斜杠
\'or'	打印一个省略号或单一引证
%%	打印一个百分号（%）

8.6.2 格式转换指定符的理解

理解大量的格式指定符最好的方法是用例子，现在我们看一些例子以及它们的结果。

8.6.2.1 情况 1：显示十进制整数数据

显示十进制整数数据要用到%d 格式转换指定符。如果需要的话，d 可能出现在标识（flag），域宽和精度指定符之前。如果有用的话，精度指定符可以指定要显示的数据的最小数字个数，如果没有足够多的数字，那么 **MATLAB** 将在这个数之前添加 0。

<table>
<tr><th>函数</th><th>结果</th><th>评论</th></tr>
<tr><td>fprintf('%d\n',123)</td><td>----|----|
123</td><td>按需要字符的个数，显示这个数据。例如数 123，需要三个字符</td></tr>
<tr><td>fprintf('%6d\n',123)</td><td>----|----|
   123</td><td>用 6 字符域宽显示数字。在这个域中的数是右对齐的。</td></tr>
<tr><td>fprintf('%6.4d\n',123)</td><td>----|----|
  0123</td><td>用 6 字符域宽显示数字，最少也要用 4 字符域宽。在这个域中的数是右对齐的。</td></tr>
<tr><td>fprintf('%-6.4d\n',123)</td><td>----|----|
0123</td><td>用 6 字符域宽显示数字。最小也要用到 4 字符域宽，在这个域中的数是左对齐的。</td></tr>
<tr><td>fprintf('%+6.4d\n',123)</td><td>----|----|
 +0123</td><td>用 6 字符域宽显示数字，最少也要用到 4 字符域宽，加上一个正/负号。在这个域中的数是右对齐的。</td></tr>
</table>

如果用格式指定符%d 显示一个非十进制数，这个指定符将会被忽略，这个数将会以科学计算法格式显示。例如

```
fprintf('%6d\n',123.4)
```

将产生结果 1.234000e+002。

8.6.2.2 情况 2：显示浮点数数据

浮点数数据的显示要用到%e，%f。，%g 格式转换指符。如果需要的话，这些格式转换指符可能出现在标识（flag），域宽和精度指定符之前。如果指定的域宽太小了，不能显示这个数，则这个域宽是无效的。否则，则应用指定的域宽。

<table>
<tr><th>函数</th><th>结果</th><th>评论</th></tr>
<tr><td>fprintf('%f\n',123.4)</td><td>----|----|
123.400000</td><td>按需要字符的个数显示这个数据。%f 默认的格式是精确到小数点后 6 位</td></tr>
<tr><td>fprintf('%8.2f\n',123.4)</td><td>----|----|
  123.40</td><td>用 8 字符域宽显示这个数，其中两域宽用于显示小数。</td></tr>
<tr><td>fprintf('%4.2f\n',123.4)</td><td>----|----|
123.40</td><td>用 6 字符域宽来显示这个数，指定的域宽因太小而被忽略。</td></tr>
<tr><td>fprintf('%10.2e\n',123.4)</td><td>----|----|
1. 23e+002</td><td>以科学记数法显示数据，域宽为 10,小数点占 2 位。默认这个数是右对齐的。</td></tr>
</table>

fprintf('%10.2e\n',123.4)	----|----| 1 .23E+002	与上面相同，只不过 E 为大写

8.6.2.3 情况 3：显示字符数据

字符数据的显示要用到%e，%c 格式转换指符。如果需要的话，这些格式转换指符可能出现在标识（flag），域宽和精度指定符之前。如果指定的域宽太小了，不能显示这个数，则这个域宽是无效的。否则，则应用指定的域宽。

函数	结果	评论
fprintf('%c\n','s')	----|----| s	显示单个字符
fprintf('%s\n','string')	----|----| string	显示一个字符串
fprintf('%8s\n','string')	----|----| string	用 8 字符串域宽显示字符串，默认是右对齐格式
fprintf('%-8s\n','string')	----|----| string	用 8 字符串域宽显示字符串，这个字符串是左对齐的

8.6.3 如何使用格式字符串

函数 fprintf 包括一个格式字符串（在要打印出的 0 或更多的值之后）。当函数 fprintf 执行时，函数 fprintf 的输出参数列表将会按格式字符串的指示输出。这个函数从从变量的左端和格式字符的左端开始执行，并从左向右扫描，输出列表的第一个值与格式字符串中第一个格式输出符联合，等等。在输出参数列表中的值必须是相同的类型，格式必须与对应的格式描述符相对应，否则的话，意外的结果将会产生。例如，假设我们要用%c 或%d 描述符显示浮点数 123。4，这个描述符将会全部被忽略，这个数将会以科学记数的方式打印出来。

编程隐患
保证 fprintf 函数中的数据类型与格式字符串中的格式转换指定符的类型要一一对应，否则将会产生意料之外的结果。

程序从左向右读取函数 fprint 中的变量列表，它也从左向右读取相应的格式字符串。按照下面的规则，程序扫描格式字符串

1. 按从左向右的顺序扫描格式字符串。

格式字符串中的第一个转换指定符与 fprint 函数输出参数列表中的第一个值相结合，依此类推。每一个格式转换指定符的类型与输出数据类型必须相同。在下面的例子中，指示符%d 与变量 a 联合，%f 与变量 b 结合，%s 与变量 c 相结合。注意指定符类型必须与数据类型相匹配。

```
a = 10; b = pi; c = 'Hello';
fprintf('Output: %d %f %s\n', a, b, c);
```

2. 在函数 fprintf 运行完所有的变量之前，如果扫描还未到达格式字符串的结尾，程序再次从头开始扫描格式字符串。例如，语句

```
a = [10 20 30 40];
fprintf('Output = %4d %4d\n',a);
```

将会产生输出

```
Output = 10 20
Output = 30 40
```

在打印完 a(2)后，函数到达格式字符串的结尾，它将会回字符串的开始打印 a(3)，a(4)

3. 如果函数 fprintf 在到达格式字符结束之前运行完所有的变量，格式字符串的应用停止在第一个格式指定符，而没有对应的变量，或者停止在格式字符串的末端。例如语句

```
a = 10; b = 15; c = 20;
fprintf('Output = %4d\nOutput = %4.1f\n', a, b, c);
```

将产生输出

```
Output =     10
Output = 15.0
Output =     20
Output = >>
```

格式字符串的应停止在%4.1f，这是它第一次与格式转换指示符不匹配。从另一方面来说，语句

```
voltage = 20;
fprintf('Voltage = %6.2f kv.\n', voltage);
```

将产生输出

```
Voltage =   20.00 kv.
```

因为它与格式转换字符串不匹配，所以格式的应用停止在格式字符串的结尾。

例 8.2 产生一个信息表

产生并打印一个数据表是说明函数 fprintf 函数就用的好方法。下面的脚本文件产生 1 到 10 中的所有整数的平方根，平方，立方，并在一个表中显示数据，并带有合适的表头。

```
% Script file: table.m
%
% Purpose:
% To create a table of square roots, squares, and
% cubes.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/20/98 S. J. Chapman Original code
%
% Define variables:
% cube              -- Cubes
% ii                -- Index variable
% square            -- Squares
% square_roots      -- Square roots
% out               -- Output array
% Print the title of the table.
fprintf(' Table of Square Roots, Squares, and Cubes\n\n');
% Print column headings
fprintf(' Number Square Root Square Cube\n');
fprintf(' ====== =========== ====== ====\n');
% Generate the required data
ii = 1:10;
square_root = sqrt(ii);
square = ii.^2;
```

```
cube = ii.^3;
% Create the output array
out = [ii' square_root' square' cube'];
% Print the data
for ii = 1:10
    fprintf (' %2d %11.4f %6d %8d\n',out(ii,:));
end
```

程序运行后，产生的结果为

```
>> table
 Table of Square Roots, Squares, and Cubes

 Number Square Root Square Cube
 ====== =========== ====== ====
   1       1.0000       1       1
   2       1.4142       4       8
   3       1.7321       9      27
   4       2.0000      16      64
   5       2.2361      25     125
   6       2.4495      36     216
   7       2.6458      49     343
   8       2.8284      64     512
   9       3.0000      81     729
  10       3.1623     100    1000
```

8.6.4 fscanf 函数

函数 fscanf 可以从一个文件中按用户自定义格式读取格式化数据。形式如下：

array = fscanf(fid, format)

[array, count] = fscanf(fid, format, size)

其中 fid 是所要读取的文件的文件标识（fileid），format 是控制如何读取的格式字符串，array 是接受数据的数组，输出参数 count 返回从文件读取的变量的个数。参数 size 指定从文件读取数据的数目。这个函数有以下三个类型。

- n　　准确地读取 n 个值。执行完相应的语句后，array 将是一个包含有 n 个值的列向量
- Inf　　读取文件中所有值。执行完相应的语句后，array 将是一个列向量，包含有从文件所有值。
- [n,m]　　从文件中精确定地读取 n×m 个值。Array 是一个 n×m 的数组。

格式字符串用于指定所要读取数据的格式。它由普通字符和格式转换指定符。函数 fscanf 把文件中的数据与文件字符串的格式转换指定符进行对比。只要两者区配，fscanf 把值进行转换并把它存储在输出数组中。这个过程直到文件结束或读取的文件当数目达到了 size 数组才会结束，无论那一种情况先出现。

如果文件中的数据与格式转换指定符不匹配，fscanf 的操作就会突然中止。

fscanf 格式转换指定符基本上与 fprintf 的格式转换指定符相同。最普通的指定符被总结在表 8.10 中。

为了说明函数 fscanf 的应用，我们将试着读取文件 x.dat，在两行中包含下面的值。

```
10.00 20.00
30.00 40.00
```

1. 如果用下面的语句读取一文件

```
[z, count] = fscanf(fid, '%f');
```

z 值为$\begin{bmatrix}10\\20\\30\\40\end{bmatrix}$.，cout 的值为 4。

2. 如果用下面的语句读取一文件

```
[z, count] = fscanf(fid, '%f', [2 2]);
```

z 的值为$\begin{bmatrix}10 & 30\\20 & 40\end{bmatrix}$，cout 的值为 4。

3. 下一步，我们让我们从一文件中读取十进制小数数据。如果用下面的语句读取一文件

```
[z, count] = fscanf(fid, '%d', Inf);
```

z 为 10，count 的值为 1。这种情况的发生是因为 10.00 的小数点与格式转义指定符不匹配，函数 fscanf 函数停止在第一次出现不匹配时。

4. 如果用下面的语句读取一文件

```
[z, count] = fscanf(fid, '%d.%d',[1 Inf]);
```

z 为行向量[10 0 20 0 30 0 40 0]，count 的值为 8。这种情况的发生是因为小数点与格式转义指定符匹配，小数点前后的数可以看作独立的整数。

5. 现在让我们文件中读取一个单独的字符，如果用下面的语句读取一文件

```
[z, count] = fscanf(fid, '%c');
```

变量 z 是一个包含文件中每一个字符的行向量，包括所有的空格和换行符!变量 count 等于文件中字符的个数。

6. 最后，让我们试着从文件中读取字符串，如果用下面的语句读取一文件

```
[z, count] = fscanf(fid, '%s');
```

z 是一个行向量，包括 20 个字符 10.0020.0030.0040.00，count 为 4。这种结果的产生是因为字符串指定符忽略空白字符，这个函数在这个文件中发现 4 个独立的字符串。

表 8.10 fscanf 的格式转化指定符

指定符	描述
%c	读取一单个字符。这个字符读取的是任意类型的字符，包括空格，换行符等
%Nc	读取 N 个字符
%d	读取一小数（忽略空格）
%e %f %g	读取一浮点数（忽略空格）
%i	读取一有符号数（忽略空格）
%a	读取一字符串。字符串可以被空格或其他类似于换行符的特殊符号隔开

8.6.5 fgetl 函数

函数 fgetl 从一文件中把下一行（最后一行除外）当作字符串来读取。它的形式为

```
line = fgetl(fid)
```

如果 fid 是我们所要读取的文件的标识（file id）。line 是接受数据的字符数组。如果函数 fgetl 遇到文件的结尾，line 的值为-1。

8.6.6 fgets 函数

函数 fgets 从一文件中把下一行（包括最后一行）当作字符串来读取。它的形式为

```
line = fgets(fid)
```

如果 fid 是我们所要读取的文件的标识（file id）。line 是接受数据的字符数组。如果函

数 fgets 遇到文件的结尾，line 的值为-1。

8.7 格式化和二进制 I/O 函数的比较

格式化 I/O 数据产生格式化文件。格式化文件夹由可组织字符，数字等组成，并以 ASCII 文本格式。这类数据很容易辨认，因为当我们把在显示器上把他显示出来，或在打印机上打印出来。但是，为了应用格式化文件中的数据，**MATLAB** 程序必须把文件中的字符转化为计算机可以直接应用的中间数据格式。格式转换指定符为这次转换提供了指令。

格式化文件有以下优点：我们可以清楚地看到文件包括什么类型的数据。它还可以非常容易在不同类型的程序间进行转换。但是也有缺点程序必须作大量的工作，对文件中的字符串进行转换，转换成相应的计算机可以直接应用的中间数据格式。如果我们读取数据到其他的 **MATLAB** 程序，所有的这些工作都会造成效率浪费。而且一个数的计算机可以直接应用的中间数据格式要比格式化文件中的数据要大得多。例如，一个 64 位浮点数的中间数据格式需要 8 个字节的内存。而格式化文件中的字符串表达形为±d.ddddddddddddddEee，它将需要 21 个字节。所以用字符格式存储数据是低效的且浪费磁盘空间。

无格式文件（**二进制文件**）克服上面的缺点，它其中的数据无需转化，就可以把内存中的数据写入磁盘。因为没有转化发生，计算机就没有时间浪费在格式化数据上。在 **MATLAB** 中，二进制 I/O 操作要比格式化 I/O 操作快得多，因为它中间没有转化。进一步说，数据占用的磁盘空间将更小。从另一方面来说，无格式的数据不能进行人工检查和人工翻译。还有，它不能移植到不同类型的计算机，因为不同类型的计算机有不同中间过程来表示整数或浮点数。

表 8.11 显示了格式化文件与无格式化文件的区别。在一般情况下，格式化文件，对于那些必须进行人工检查的数据，或对于那些必须在不同的计算机上运行的数据，是最好的选择。对于那些不需要进行人工检查的数据且在相同类型的计算机创建并运行的数据，存储最好用无格式文件。在这些环境下，无格式文件运算要快得多，占用的磁盘空间更小。

好的编程习惯

对于那些必须进行人工检查的数据，或对于那些必须在不同的计算机上运行的数据，用格式化文件创建数据。对于那些不需要进行人工检查的数据且在相同类型的计算机创建并运行的数据，用无格式文件创建数据，当 I/O 速度缓慢时，用格式化文件创建数组。

表 8.11 格式化文件和无格式化文件的比较

格式化文件	无格式化文件
能在输出设备显示数据	不能在输出设备显示数据
能在不同的计算机上很容易地进行移植	不能在不同的计算机上很容易地进行移植
相对地，需要大量的磁盘空间	相对地，需要较少的磁盘空间
慢：需要大量的计算时间	快：需要较少的计算时间
在进行格式化的过程中，产生截断误差或四舍五入错误	不会产生截断误差或四舍五入错误

例 8.3 格式化和二进制 I/O 文件的比较

在这个例子中的程序比较了用格式化和二进制 I/O 操作读写一个含 10000 个元素数组所花的时间。注意每一个操作运行 10 次求平均值。

```
% Script file: compare.m
%
% Purpose:
```

```
% To compare binary and formatted I/O operations.
% This program generates an array of 10,000 random
% values and writes it to disk both as a binary and
% as a formatted file.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/19/98 S. J. Chapman Original code
%
% Define variables:
% count          -- Number of values read / written
% fid            -- File id
% in_array       -- Input array
% msg            -- Open error message
% out_array      -- Output array
% status         -- Operation status
% time           -- Elapsed time in seconds
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate the data array.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
out_array = randn(1,10000);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First, time the binary output operation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reset timer
tic;
% Loop for 10 times
for ii = 1:10
    % Open the binary output file for writing.
    [fid,msg] = fopen('unformatted.dat','w');
    % Write the data
    count = fwrite(fid,out_array,'float64');
    % Close the file
    status = fclose(fid);
end
% Get the average time
time = toc / 10;
fprintf ('Write time for unformatted file = %6.3f\n',time);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Next, time the formatted output operation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reset timer
tic;
% Loop for 10 times
for ii = 1:10
    % Open the formatted output file for writing.
    [fid,msg] = fopen('formatted.dat','wt');
    % Write the data
    count = fprintf(fid,'%23.15e\n',out_array);
    % Close the file
    status = fclose(fid);
end
% Get the average time
time = toc / 10;
fprintf ('Write time for formatted file = %6.3f\n',time);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Time the binary input operation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reset timer
tic;
% Loop for 10 times
for ii = 1:10
    % Open the binary file for reading.
    [fid,msg] = fopen('unformatted.dat','r');
    % Read the data
    [in_array, count] = fread(fid,Inf,'float64');
    % Close the file
    status = fclose(fid);
end
% Get the average time
time = toc / 10;
fprintf ('Read time for unformatted file = %6.3f\n',time);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Time the formatted input operation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reset timer
tic;
% Loop for 10 times
for ii = 1:10
    % Open the formatted file for reading.
    [fid,msg] = fopen('formatted.dat','rt');
    % Read the data
    [in_array, count] = fscanf(fid,'%f',Inf);
    % Close the file
    status = fclose(fid);
end
% Get the average time
time = toc / 10;
fprintf ('Read time for formatted file = %6.3f\n',time)
```

当程序在奔腾 733MHz 机器上运行，操作系统为 windowsNT2000 专业版，得到的结果为

```
>> compare
Write time for unformatted file =   0.002
Write time for formatted file =   0.132
Read time for unformatted file =   0.002
Read time for formatted file =   0.157
```

写入磁盘的文件如下所示：

```
D:\MATLAB\work\chap8>dir
 驱动器 D 中的卷是 SINSTALL
 卷的序列号是 C33D-3233

 D:\MATLAB\work\chap8 的目录

2008-01-17   18:58     <DIR>          .
2008-01-17   18:58     <DIR>          ..
2008-01-17   18:51            250,000 formatted.dat
2008-01-17   18:51             80,000 unformatted.dat
               2 个文件        330,000 字节
               2 个目录  2,495,549,440 可用字节
```

注意写入格式化文件数据所需的时间是无格式文件的 60 倍，记取时间是无格式文件的 75 倍。还有格式化文件的大小是无格式文件的 3 倍。得到的结果是非常清楚的，除非你真

得需要格式化数据，否则二进制 I/O 操作是 **MATLAB** 中存储数据的一个非常好的方法。

测试 8.2

本测试提供了一个快速的检查方式，看你是否掌握了 8.1 到 8.5 的基本内容。如果你对本测试有疑问，你可以重读 8.1 到 8.5，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. 格式化和二进制 I/O 操作的区别是什么？
2. 什么时候我们应当用格式化 I/O 操作？什么时候我们应当有二进制 I/O 操作？
3. 编写 **MATLAB** 语句创建一个表，由 x 的正弦值和余弦值（x = 0, 0.1pi, ..., pi），在表上有标题和标签。

看第 4 题和第 5 题，判断 **MATLAB** 语句是否正确。如果有错误，指出错在那里。

4.
```
a = 2*pi;
b = 6;
c = 'hello';
fprintf(fid, '%s %d %g\n',a, b, c);
```
5.
```
data1 = 1:20;
data2 = 1:20;
fid = fopen('xxx', 'w+');
fwrite(fid, data1);
fprintf(fid, '%g\n', data2);
```

8.8 文件位置和状态函数

正如我们前面所陈述的，**MATLAB** 文件是连续的——它们从第一条记录开始一直读到最后一条记录。但是，有时在一个程序中，我们需要多次调用一段数据或整个文件。在一个连续文件中，我们如何跳过无用的数据呢？

在打开文件之前，**MATLAB** 函数 exist 用于判断这个文件是否存在。一旦一个文件打开，我们就可以用函数 feof 和 ftell 判断当前数据在文件中的位置。还用两个函数帮助我们在文件中移动：frewind 和 fseek。

最后，当程序发生 I/O 错误时，**MATLAB** 函数 ferror 将会对这个错误进行详尽的描述。我们现在将向大家详细的介绍这 6 个函数，我们先看一下 ferror，因为它可以应用其他的函数中。

8.8.1 exist 函数

exist 函数用来检测工作区中的变量，内建函数或 **MATLAB** 搜索路径中的文件是否存在。它的形式如下

```
ident = exist('item');
ident = exist('item', 'kind');
```

如果“item”存在，函数就根据它的类型返回一个值。可能的结果被显示在表 8.12 中。

函数 exist 指定所要搜索的条目（item）的类型。它的合法类型为“var”，“file”，“builtin”和“dir”。

函数 exist 是非常重要的，因为我们可以利用它判断一个文件否存在。当文件被打开时，fopen 函数中权限运算符“w”和“w+”会删除文件已有的一个文件。在程序员允许 fopen 函数删除一个文件时，它必须征得用户的同意。

表 8.12 由函数 exist 的返回值

值	意义
0	没有发现条目
1	条目为当前工作区的一个变量
2	条目为 m 文件或未知类型的文件
3	条目是一个 MEX 文件
4	条目是一个 MDL 文件
5	条目是一个内建函数
6	条目是一个 p 代码文件
7	条目是一个目录

例 8.4 打开一个输出文件

这个程序从用户那里得到输出文件名，并检查它是否存在。如果存在，就询问用户是要把用新数据覆盖这个文件，还是要把新的数据添加到这个文件中。如果这个文件不存在，那么这个程序就会很容易地打开输出文件。

```
% Script file: output.m
%
% Purpose:
% To demonstrate opening an output file properly.
% This program checks for the existence of an output
% file. If it exists, the program checks to see if
% the old file should be deleted, or if the new data
% should be appended to the old file.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 11/29/98 S. J. Chapman Original code
%
% Define variables:
% fid -- File id
% out_filename -- Output file name
% yn -- Yes/No response
% Get the output file name.
out_filename = input('Enter output filename: ','s');
% Check to see if the file exists.
if exist(out_filename,'file')
    % The file exists
    disp('Output file already exists.');
    yn = input('Keep existing file? (y/n) ','s');
    if yn == 'n'
        fid = fopen(out_filename,'wt');
    else
        fid = fopen(out_filename,'at');
    end
else
    % File doesn't exist
    fid = fopen(out_filename,'wt');
end
% Output data
fprintf(fid,'%s\n',date);
% Close file
```

```
fclose(fid);
```

当这个程序执行后，产生的结果为

```
>> output
Enter output filename: xxx
>> type xxx
17-Jan-2008
>> output
Enter output filename: xxx
Output file already exists.
Keep existing file? (y/n) y
>> type xxx
17-Jan-2008
17-Jan-2008
>> output
Enter output filename: xxx
Output file already exists.
Keep existing file? (y/n) n
>> type xxx

17-Jan-2008
```

三种不同的情况均产生了正确的结果。

好的编程习惯
未经用户同意，不要用新数据覆盖原用的文件。

8.8.2 函数 ferror

在 **MATLAB** 的 I/O 系统中有许多的中间数据变量，包括一些专门提示与每一个打开文件相关的错误的变量。每进行一次 I/O 操作，这些错误提示就会被更新一次。函数 ferror 得到这些错误提示变量，并把它转化为易于理解的字符信息。

```
message = ferror(fid)
message = ferror(fid, 'clear')
[message, errnum] = ferror(fid)
```

这个函数会返回与 fid 相对应文件的大部分错误信息。它能在 I/O 操作进行后，随时被调用，用来得到错误的详细描述。如果这个文件被成功调用，产生的信息为“…”，错误数为 0。

对于特殊的文件标识，参数“clear”用于清除错误提示。

8.8.3 函数 foef

函数 feof 用于检测当前文件的位置是否是文件的结尾。它的形式如下

```
eofstat = feof(fid)
```

如果是文件的结尾，那么函数返回 1，否则返回 0。

8.8.4 函数 ftell

函数 ftell 返回 fid 对应的文件指针读/写的位置。这个位置是一个非负整数，以 byte 为

单位，从文件的开头开始计数。返回值-1 代表位置询问不成功。如果这种情况发生了，我们利用 ferror 得知为什么询问不成功。函数的形式如下：

```
position = ftell(fid)
```

8.8.5 函数 frewind

函数 frewind 允许程序把文件指针复位到文件的开头，形式如下

```
frewind(fid)
```

这个函数不返回任何状态信息。

8.8.6 函数 fseek

函数 fseek 允许程序把文件指针指向文件中任意的一个位置。函数形式如下

```
status = fseek(fid, offset, origin)
```

函数用 offsett 和 origin 来重设 fid 对应文件的文件指针。offset 以字节为单位，带有一个正数，用于指向文件的结尾，带有一个负数，用于指向文件的开头。origin 是一个字符串，取值为下面三个中的一个。

- “bof” 文件的开始位置
- “cof” 指针中的当前位置
- “eof” 文件的结束位置

如果这个操作成功，status 的值为 0，如果操作失败 status 为-1。如果 status 为-1，用函数 ferror 判断错误出现在那里。

举一个例子，用于说明 fseek 和 ferror 的联合应用，考虑下面的语句。

```
[fid, msg] = fopen('x', 'r');
status = fseek(fid, -10, 'bof');
if status ~= 0
    msg = ferror(fid);
    disp(msg);
end
```

这些命令打开了一个文件，并把文件指针设置在文件开始之前的 10 个字节上面。这是不可能，所以 fseek 将会返回一个-1，用 ferror 得到对应的错误信息。当这些语句被编译时，产生下面的错误信息。

```
Offset is bad - before beginning-of-file.
```

例 8.5

用一系列带有噪声的值拟合一条直线

在例 4.7 中，我们学习了用最小二乘法拟合直线

$$y = mx + b \qquad (8.1)$$

确定待定系数 m 和 b 的标准方法为最小二乘法。之所以称为最小二乘法，是因为根据偏差的平方和为最小的条件来选择常数 m 和 b 的。公式如下:

$$m = \frac{(\Sigma xy)-(\Sigma x)\bar{y}}{(\Sigma x^2)-(\Sigma x)\bar{x}} \qquad (8.2)$$

$$b = \bar{y}-m\bar{x} \qquad (8.3)$$

其中，$\sum x$ 代表所有测量值 x 之和，$\sum y$ 代表所有测量值 y 之和，$\sum xy$ 代表所有对应的 x 与 y 的乘积之和，$\bar{x}$ 代表测值量 x 的数学期望，$\bar{y}$ 代表测值量 y 的数学期望。

编写一个程序，用最小二乘法计算出 m 和 b。其中有一系列含有噪声的数据(x,y)存储在输入数据文件中。

答案

1. 陈述问题

用最小二乘法计算直线的截距 b 和斜率 m，输入值任意数目的（x，y）坐标对。输入值（x，y）被存储在用户自定义输入文件中。

2. 定义输入输出值

程序所需的输入值是（x，y）坐标对，x，y 均为实数，每一个点都存储在磁盘文件独立的行中，输出是用最小二乘法计算出的直线的截距 b 和斜率 m。

3. 算法描述

本程序可以分为以下四大步骤

```
Get the name of the input file and open it
Accumulate the input statistics
Calculate the slope and intercept
Write out the slope and intercept
```

这个程序的第一大步得到输入文件的名字并打开这个文件。为了做到这一点，我们必须提示用户键入输入文件名。在文件打开之后，我们必须检查文件是否被成功打开。下一步，我们必须从文件中读取数据，并记录下这些数据，并计算$\sum x$，$\sum y$，$\sum x^2$ 和$\sum xy$。这些步骤的伪代码如下：

```
Initialize n, sum_x, sum_x2, sum_y, and sum_xy to 0
Prompt user for input file name
Open file 'filename'
Check for error on open
if no error
    READ x, y from file 'filename'
    while not at end-of-file
        n ← n + 1
        sum_x ← sum_x + x
        sum_y ← sum_y + y
        sum_x2 ← sum_x2 + x^2
        sum_xy ← sum_xy + x*y
        READ x, y from file 'filename'
    end
    (further processing)
end
```

下一步我们要用最小二乘法计算直线的截距 b 和斜率 m。我们可以根据公式（8.2）（8.3）得到。

```
x_bar ← sum_x / n
y_bar ← sum_y / n
slop ← (sum_xy - sum_x*y_bar) / (sum_x2 - sum_x*x_bar)
y_int ← y_bar - slop * x_bar
```

最后，我们要写出结果。

```
Write out slope 'slope' and intercept 'y_int'.
```

4. 转化为 **MATLAB** 语言

```
% Script file: lsqfit.m
%
% Purpose:
% To perform a least-squares fit of an input data set
% to a straight line, and print out the resulting slope
% and intercept values. The input data for this fit
```

```
% comes from a user-specified input data file.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 12/20/98 S. J. Chapman Original code
%
% Define variables:
% count -- number of values read
% filename -- Input file name
% fid -- File id
% msg -- Open error message
% n -- Number of input data pairs (x,y)
% slope -- Slope of the line
% sum_x -- Sum of all input X values
% sum_x2 -- Sum of all input X values squared
% sum_xy -- Sum of all input X*Y values
% sum_y -- Sum of all input Y values
% x -- An input X value
% x_bar -- Average X value
% y -- An input Y value
% y_bar -- Average Y value
% y_int -- Y-axis intercept of the line
% Initialize sums
n = 0; sum_x = 0; sum_y = 0; sum_x2 = 0; sum_xy = 0;
% Prompt user and get the name of the input file.
disp('This program performs a least-squares fit of an');
disp('input data set to a straight line. Enter the name');
disp('of the file containing the input (x,y) pairs: ' );
filename = input(' ','s');
% Open the input file
[fid,msg] = fopen(filename,'rt');
% Check to see if the open failed.
if fid < 0
    % There was an error--tell user.
    disp(msg);
else
    % File opened successfully. Read the (x,y) pairs from
    % the input file. Get first (x,y) pair before the
    % loop starts.
    [in,count] = fscanf(fid,'%g %g',2);
    while ~feof(fid)
        x = in(1);
        y = in(2);
        n = n + 1; %
        sum_x = sum_x + x; % Calculate
        sum_y = sum_y + y; % statistics
        sum_x2 = sum_x2 + x.^2; %
        sum_xy = sum_xy + x * y; %
        % Get next (x,y) pair
        [in,count] = fscanf(fid,'%f',[1 2]);
    end
    % Now calculate the slope and intercept.
    x_bar = sum_x / n;
    y_bar = sum_y / n;
    slope = (sum_xy - sum_x*y_bar) / (sum_x2 - sum_x*x_bar);
    y_int = y_bar - slope * x_bar;
```

```
        % Tell user.
        fprintf('Regression coefficients for the least-squares line:\n');
        fprintf(' Slope (m) = %12.3f\n',slope);
        fprintf(' Intercept (b) = %12.3f\n',y_int);
        fprintf(' No of points = %12d\n',n);
    end
```

5. 检测程序

为了检测这个程序，我们可以用一些简单的数据集合进行检测。例如，如果输入数据所对应的点都在同一条直线，那么产生的斜率和截距必定是那条直线的斜率和截距。这组数据为

```
1.1   1.1
2.2   2.2
3.3   3.3
4.4   4.4
5.5   5.5
6.6   6.6
7.7   7.7
```

它的斜率和截距分别为 1.0 和 0.0。我们把这些值写入 input1 文件，运行这个程序，结果如下:

```
>> lsqfit
This program performs a least-squares fit of an
input data set to a straight line. Enter the name
of the file containing the input (x,y) pairs:
 input1.txt
Regression coefficients for the least-squares line:
 Slope (m) =           1.000
 Intercept (b) =           0.000
 No of points =               7
```

我们在这些测量值上加入一些噪声，数据变为

```
1.1   1.01
2.2   2.30
3.3   3.05
4.4   4.28
5.5   5.75
6.6   6.48
7.7   7.84
```

如果把这些值写入 input2 文件，运行程序，结果如下

```
>> lsqfit
This program performs a least-squares fit of an
input data set to a straight line. Enter the name
of the file containing the input (x,y) pairs:
 input2.txt
Regression coefficients for the least-squares line:
 Slope (m) =           1.024
 Intercept (b) =          -0.120
 No of points =               7
```

我们用手动计算很容易就能得到上面两个程序的正确结果。第二个程序图象如图 8.2 所示。

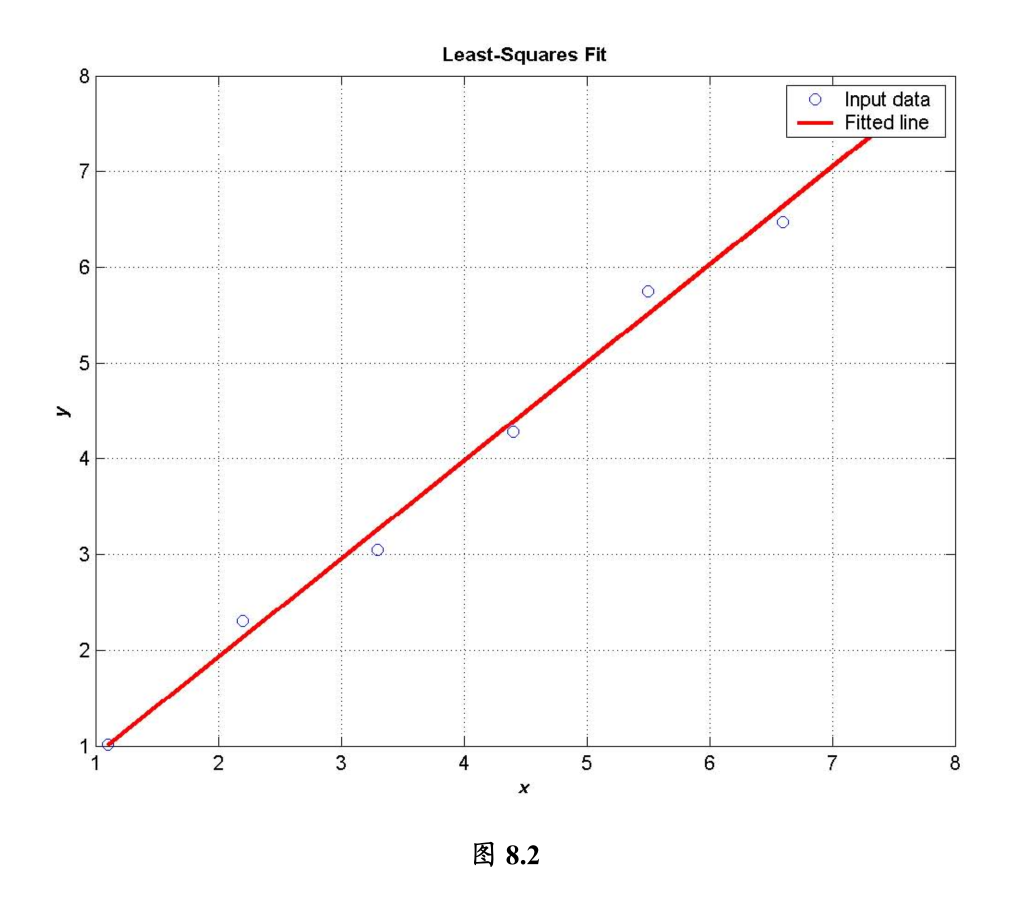


图 8.2

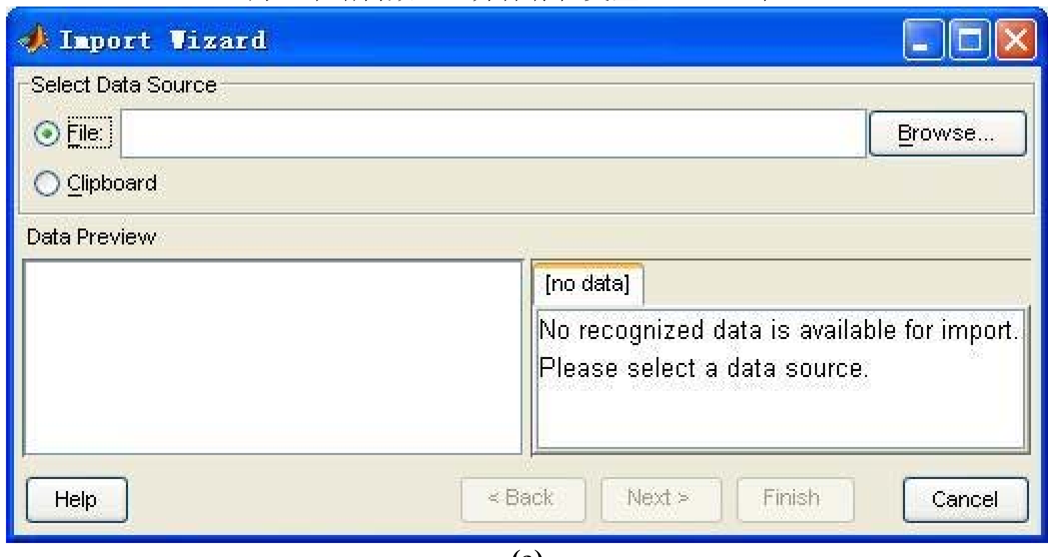
8.9 函数 uiimport

函数是一种基于 GUI 的方法从一个文件或从剪贴板中获取数据。这个命令的形式如下

uiimport

structure = uiimport;

在第一种情况下，获取的数据可以直接插入当前的工作区。在第二种情况下，数据可以转化为一个结构数组。并保存在变量 structure 中。



(a)

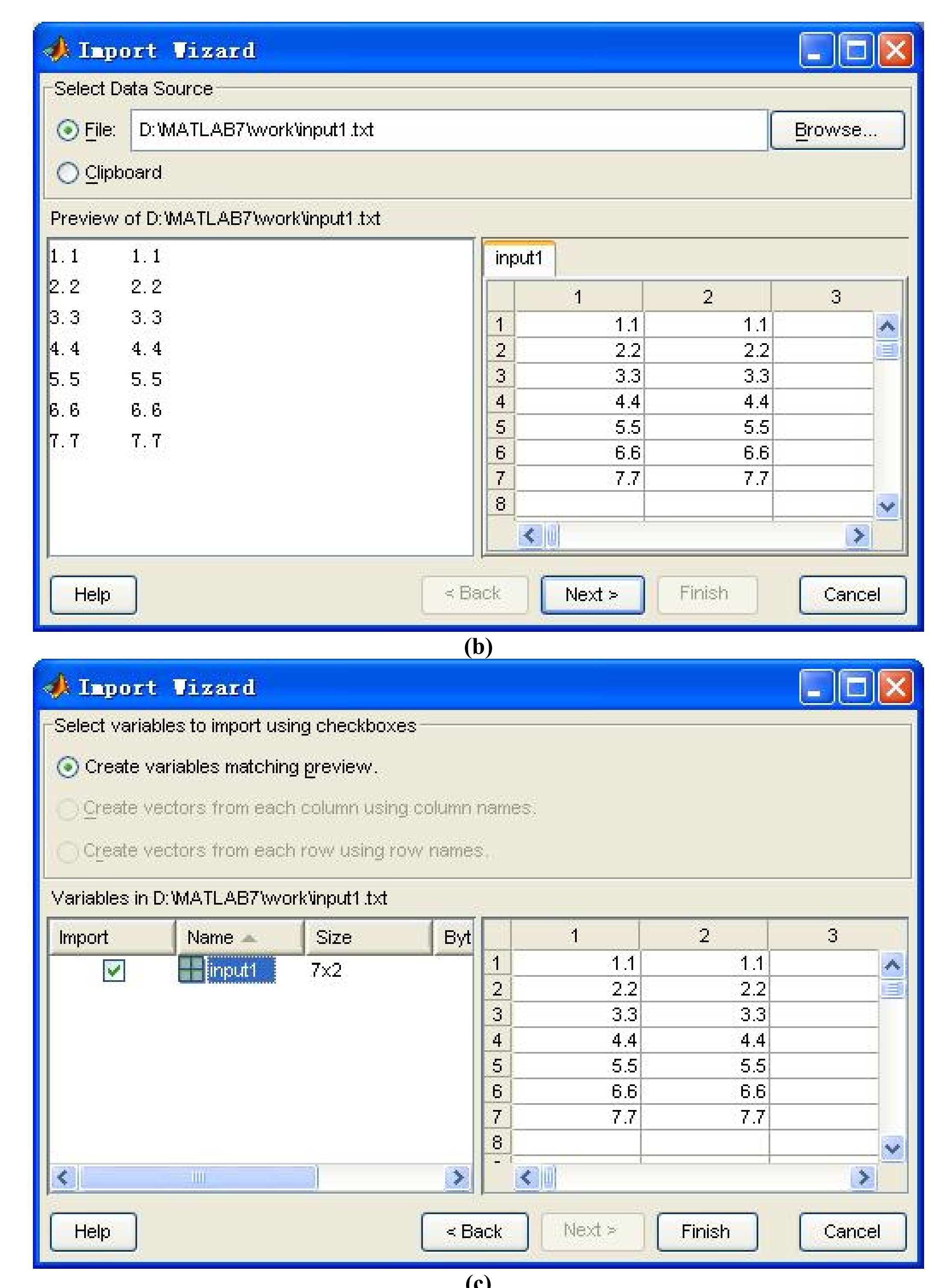


图 8.5 uiimport 的应用（a）导入向导 (b)选择一个数据文件后，创建一个或多个数组，它的内容可以被检测。(c)一旦一个文件被加载后，用户可以选择哪些数组要导入到 MATLAB 中。

当我们在命令窗口中键入命令 uiimport，importwizard 将会以窗口的形式显示出来。用户可以选择获取数据的文件或剪贴板。它支持许多的不同格式，它可以读取任意应用程序保存在剪贴板是的数据。当你想要把数据导入 **MATLAB** 并对其进行分析时，它的灵活性将会非常地有用。

8.10 总结

8.10.1 好的编程习惯总结

1. 除非我们必须与非 **MATLAB** 程序进行数据交换，存储和加载文件时，都应用 mat 文件格式。这种格式是高效的且移植性强，它保存了所有 **MATLAB** 数据类型的细节。

2. 在使用 fopen 语句时，一定要注意指定合适的权限，这取决于你是要读取数据，还是要写入数据。好的编程习惯可以帮助你避免（类似于覆盖的）错误。

3. 在文件打开操作后检查它的状态以确保它被成功打开。如果文件打开失败，提示用户解决方法。

4. 对于那些必须进行人工检查的数据，或对于那些必须在不同的计算机上运行的数据，用格式化文件创建数据。对于那些不需要进行人工检查的数据且在相同类型的计算机创建并运行的数据，用无格式文件创建数据，当 I/O 速度缓慢时，用格式化文件创建数组。

5. 未经用户同意，不要用新数据覆盖原用的文件。

8.10.2 MATLAB 总结

函数与命令

MATLAB 输入/输出语句

类别	函数	描述
加载/保存工作区	load	加载工作区
	save	保存工作区
文件打开/关闭	fopen	打开文件
	fclose	关闭文件
二进制 I/O	fread	从文件中读取二进制数据
	fwrite	把二进制数据写入文件
格式化 I/O	fscanf	从文件中读取格式化数据
	fprintf	把格式化数据写入文件
	fgetl	读取文件的一行，忽略换行符
	fgets	读取文件的一行，不忽略换行符
文件位置、状态	delete	删除文件
	exist	检查文件是否存在
	ferror	所需文件的 I/O 错误情况
	feof	检测文件的结尾
	fseek	设置文件的位置
	ftell	检查文件的位置
	frewind	回溯文件
临时情况	tempdir	得到临时目录名
	tempname	得到临时文件名

8.11 练习

8.1 二进制 I/O 与格式化 I/O 有什么区别？哪一个函数用于显示 I/O 操作的类型？

8.2 对数表。编写一个程序，能够产生 x 以十为底的对数，x 的取值范围为 x1:0.1:10，表应该在新的一页开始，表应该有标题以及用来说明行与列的标签。这个表如下所示

	x.0	x.1	x.2	x.3	x.4	x.5	x.6	x.7	x.8	x.9
1.0	0.000	0.041	0.079	0.114	...					
2.0	0.301	0.322	0.342	0.362	...					
3.0	...									
4.0	...									
5.0	...									
6.0	...									
7.0	...									
8.0	...									
9.0	...									
10.0	...									

8.3 编写一个 **MATLAB** 程序，从一天的开始，每一秒读取一次时间（这个值应在 0.0 到 86400.0 之间）并打印出相应的时间 HH:MM:SS(以 24 时进制计时)用合适的格式转换符，把前面的零存储在 MM 和 SS 域。确保输入的秒数是合法的，如果输入了不合法的数据则提供一个合适的错误信息。

8.4 重力加速度。重力加速度 g 可由下面的公式计算得出

$$g = -G\frac{M}{(R+h)^2} \qquad (8.4)$$

编写一个程序，h 的取值为 0:500:40000（单位千米），计算出相应的重力加速度，打印出一个高度- 加速度表，并包括所有的输出变量。画出这些数据的图象。

8.5 在例 8.5 中的程序说明了格式化 I/O 命令从磁盘中读取（x，y）数据的应用。我们还可以用函数 load -sacii。重新编写程序应用函数 load -sacii。检测你重新编写的程序，确认它产生一个与例 8.5 相同的结果。

8.6 用函数 textread 重新编写例 8.5 中的程序。比较 8.4，8.5 和 8.6，看那一个函数更容易使用。

8.7 编写程序，从用户自定义的输入文件中读取任意数目的实数，对这些数进行四舍五入，然后把得到的整数值写到用户自定义输出文件中。确保输入文件存在，如果它不存在，通知用户并询问是否有其他的输入文件。如果输出文件存在，询问是否删除它。如果不删除，那么提示用另一个输出文件名。

8.8 正弦表和余弦表。编写一程序，产生一个 θ 正弦|余弦表 θ=0:1:90（单位为度）。程序应当合适的行标和列标。

8.9 本利计算。假设你在银行总共的存款为 P（p 代表本金）。如果银行的年利率为 i%（月利率则是年利率的 12 分之 1），那么存入银行 n 个月后，得到的本金 F 为

$$F = P\left(1 + \frac{i}{1200}\right)^n \qquad (8.5)$$

编写一个程序，读取本金 P 和年利率 i，并计算出五年内每一个月的本金 F，并列出一个表。这个表被写入输出文件“interest”。确保表中的行有合适的标签。

8.10 编写一个程序从输入数据文件中读取一系列的整数，并找出这个文件中的最大值与最小值。并打印出来这两个值，和它所在的行。假设你不知道输入文件中数据的个数。

8.11 平均数。修改练习 4.27 中程序。从一个输入数据文件中读取任意数目的值，并计算出相应的算术平均数，几何平均数，均方根平均数，调和均数。为检测这个程序。输入数据文件中包含下面的数据

1.0, 2.0, 5.0, 4.0, 3.0, 2.1, 4.7, 3.0

8.12 把角度的弧度格式转换为相应的度/分/秒。

编写一程序，从磁盘文件中读取弧度格式的角度，并把它转化为相应的度/分/秒格式。为检测这个程序，磁盘文件中应包含下面的数据

0.0, 1.0, 3.141593, 6.0

8.13 在你的计算机中用其他程序创建一系列数据，例如，word，excel 或记事本。把这些数据复制到 window 的剪贴板上，然后用函数 uiimport 加载这些数据到 **MATLAB** 中。

第九章 句柄图形

句柄图形是对底层图形函数集合的总称，它实际上进行生成图形的工作。这些函数一般隐藏于 M 文件内部，但是它们非常地重要，因为程序员可以利用它对图象或图片的外观进行控制。例如，我们可以利用句柄图形只对 x 轴产生网格线，或选择曲线的颜色为桔黄色，桔黄色 plot 命令中的标准 LineSpec 参数。还有，句柄图形可以帮助程序员为他们的程序创建用户图形界面，用户图形界面，我们将在下一章介绍。

在本章中，我们向大家介绍 **MATLAB** 图形系统的结构，以及如何控制图形对象的属性。

9.1 MATLAB 图形系统

MATLAB 图形系统是建立图形对象的等级系统之上，每一个图形对象都有一个独立的名字，这个名字叫做句柄。每一个图形对象都有它的属性，我们可以通过修改它的属性来修改物体的行为。例如，一条曲线是图形对象的一种。曲线对象有以下的属性:x 数据，y 数据，颜色，线的类型，线宽，符号类型等等。修改其中的一个属性就会改变图象窗口中的一个图象。

由图形命令产生的每一件东西部是图形对象。例如，图形中的每一个曲线，坐标轴和字符串是独立的对象(拥有独立的名字句柄，还有形式)。所有的图象对象按**子对象**和**父对象**的形式管理，如图 9.1 所示。当一个子对象被创建时，它可能继承了父对象的许多属性。

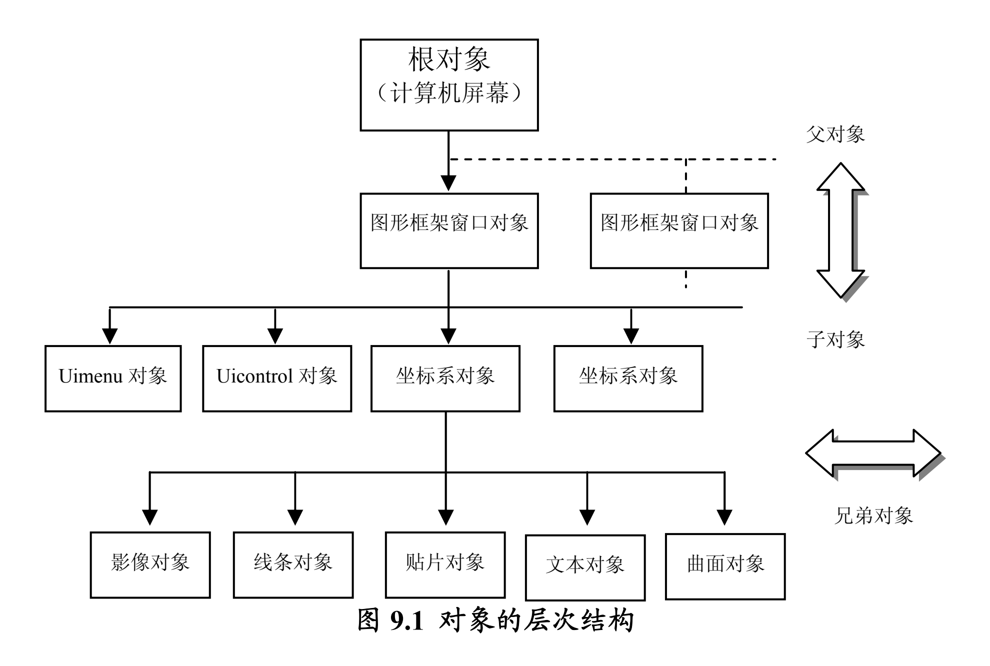


图 9.1 对象的层次结构

在 **MATLAB** 中最高层次的图形对象被根对象，我们可以通过它对整个计算机屏幕进行控制。当 **MATLAB** 启动时，根对象会被自动创建，它一直存在到 **MATLAB** 关闭。与根对象相关的属性是应用于所用 **MATLAB** 窗口的默认属性。

在根对象下，有多个图象窗口，或只有图象。每一个图象在用于显示图象数据的计算机屏幕上都有一个独立的窗口，每一个图象都有它独立的属性。与图象相关的属性有，颜色，图片底色，纸张大小，纸张排列方向，指针类型等。

每一个图形可包括四个对象:Uimenu 对象，Uicontrol 对象，坐标系对象和 Uicontextmenus 对象。Uimenu 对象，Uicontrol 对象，和 Uicontextmenus 对象是专门地用来创建用户图形界面的对象，它们将在下一章讨论。坐标系对象是指在用于显示图象的图片中的区域。在一个图象窗口中，它可能含有一个或多个坐标系。

每一个坐标系对象可能包括曲线对象，文本对象，贴片对象，还有其他的你所需的图形对象。

9.2 对象句柄

每一个图象对象都有一个独一无二的名字，这个名字叫做句柄。句柄是在 **MATLAB** 中的一个独一无二的整数或实数，用于指定对象的身份。用于创建一个图象对象的任意命令都会自动地返回一个句柄。例如，命令

```
>>Hnd1 = figure;
```

创建一个新的图象，并返回这个图象的句柄到变量 Hnd1。根对象句柄一般为 0，图象(图)对象的句柄一般是一个小的正整数，例如 1，2，3……而其他的图形(graphic)对象为任意的浮点数。

我们可以利用 **MATLAB** 函数得到图象，坐标系和其他对象的句柄。例如，函数 gcf 返回当前

图象窗口的句柄，而函数 gca 则返回在当前图象窗口中的当前坐标系对象的句柄，函数 gco 返回当前选择对象的句柄。这些函数将会在后面将会被具体讨论。

为了方便，存储句柄的变量名要在小写字母后面个 H。这样就可以与普通变量(所有的小写变量，大写变量，全局变量)区分开来。

9.3 对象属性的检测和更

对象属性是一些特殊值，它可以控制对象行为的某些方面。每一个属性都有一个属性名和属性值。属性名是用大小写混合格式写成的字符串，属性名中的每一个单词的第一个字母为大写，但是 **MATLAB** 中的变量名的大小不与区分。

9.3.1 在创建对象时改变对象的属性

当一个对象被创建时，所有的属性都会自动初始化为默认值。包含有"propertyname(属性名)"的创建函数创建对象时，默认值会被跳过，而跳过的值在创建函数中有。例如，我们在第二章看到，线宽属性可以通过下面的 plot 命令改变。

```
plot(x, y, 'LineWidth', 2);
```

录一个曲线被创建时，函数用值 2 来替代它的默认值。

9.3.2 对象创建后改变对象的属性

我们可以用随时用 get 函数检测任意一个对象的属性，并用 set 函数对它进行修改。get 函数最常见的形式如下

```
value = get(handle, 'PropertyName');
value = get(handle);
```

value 是勤句柄指定对象的属性值。如果在调用函数时，只有一个句柄，那么函数将会

返

回一个结构，域名为这个对象的属性名，域值为属性值。

set 函数的最常用形式为

```
set(handle,'PropertyName1', value1, ...);
```

在一个单个的函数中可能有多个"propertyname"和"value"。

例如，假设我们用下面的语句，画出函数 $y(x)= x^2$ 在(0，2)中的图象

```
x = 0:0.1:2;
y = x .^2;
Hnd1 = plot(x, y);
```

图象如图 9.2a 所示。这个曲线的句柄被存储在变量 Hnd1 内，我们可以利用它检测和修改这条曲线的属性。函数 get(Hnd1)在一个结构中返回这条曲线所有的属性，每一个属性名都为结构的一个元素。

```
>> result=get(Hnd1)

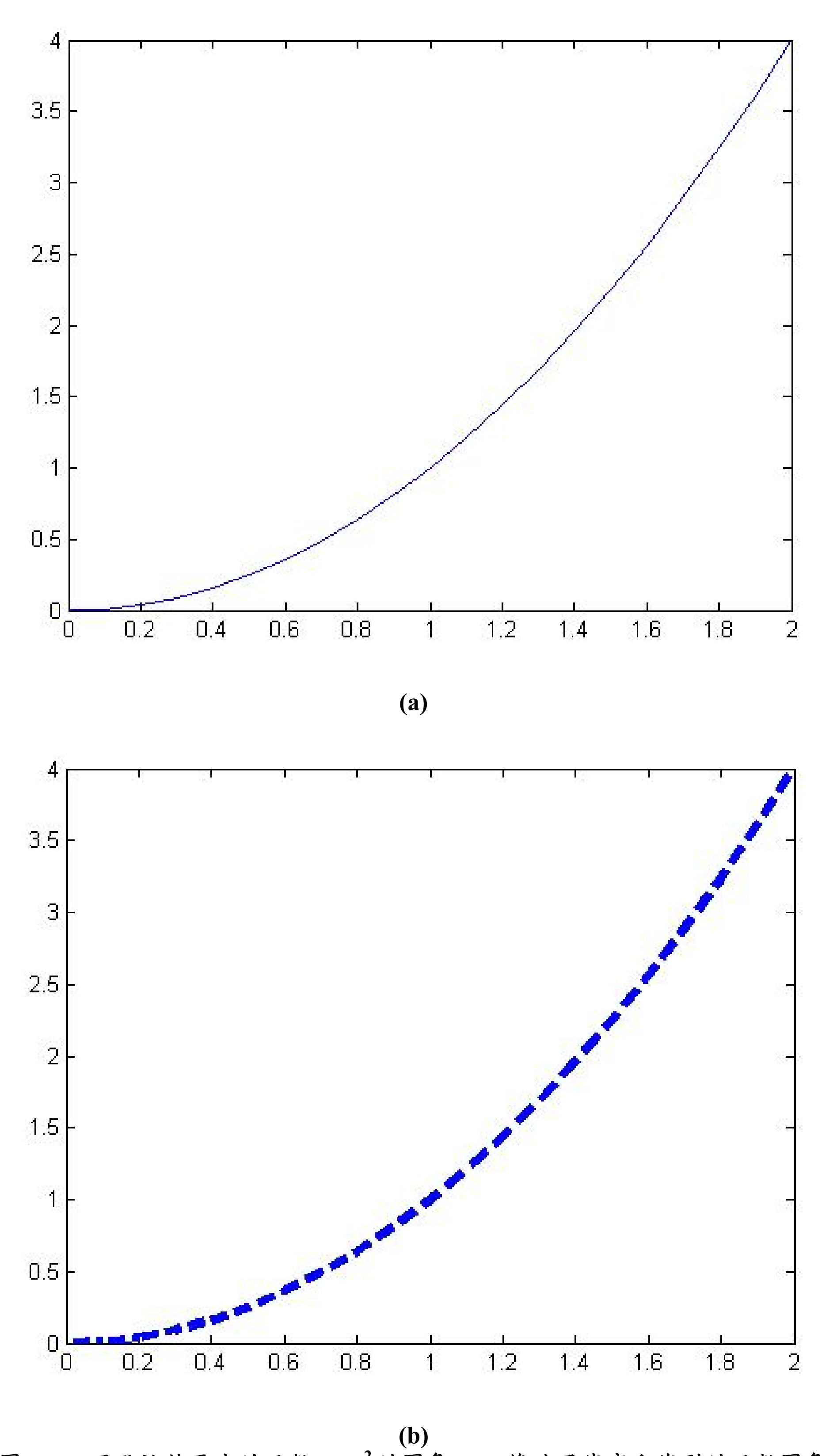
result =

                   Color: [0 0 1]
               EraseMode: 'normal'
               LineStyle: '-'
               LineWidth: 0.5000
                  Marker: 'none'
              MarkerSize: 6
         MarkerEdgeColor: 'auto'
         MarkerFaceColor: 'none'
                   XData: [1x21 double]
                   YData: [1x21 double]
                   ZData: [1x0 double]
            BeingDeleted: 'off'
           ButtonDownFcn: []
                Children: [0x1 double]
                Clipping: 'on'
               CreateFcn: []
               DeleteFcn: []
              BusyAction: 'queue'
        HandleVisibility: 'on'
                 HitTest: 'on'
           Interruptible: 'on'
                Selected: 'off'
      SelectionHighlight: 'on'
                     Tag: ''
                    Type: 'line'
           UIContextMenu: []
                UserData: []
                 Visible: 'on'
                  Parent: 151.0012
             DisplayName: ''
               XDataMode: 'manual'
             XDataSource: ''
             YDataSource: ''
             ZDataSource: ''
```

注意当前曲线的线宽为 0.5pixel，线型为虚线。我们能够用这些命令改变线型和线宽。

```
>>set(Hnd1,'LineWidth',4,'LineStyle','--')
```

产生的结果图象如 9.2b 所示。



(a)

(b)

图 9.1(a)用默认值画出的函数 $y= x^2$ 的图象，(b)修改了线宽和线型的函数图象

函数 get 和 set 对程序员来说非常的有用，因为它们可以直接插入到 **MATLAB** 程序中，根据用户的输入修改图象。在下一章，我们把它用于 GUI 编程。

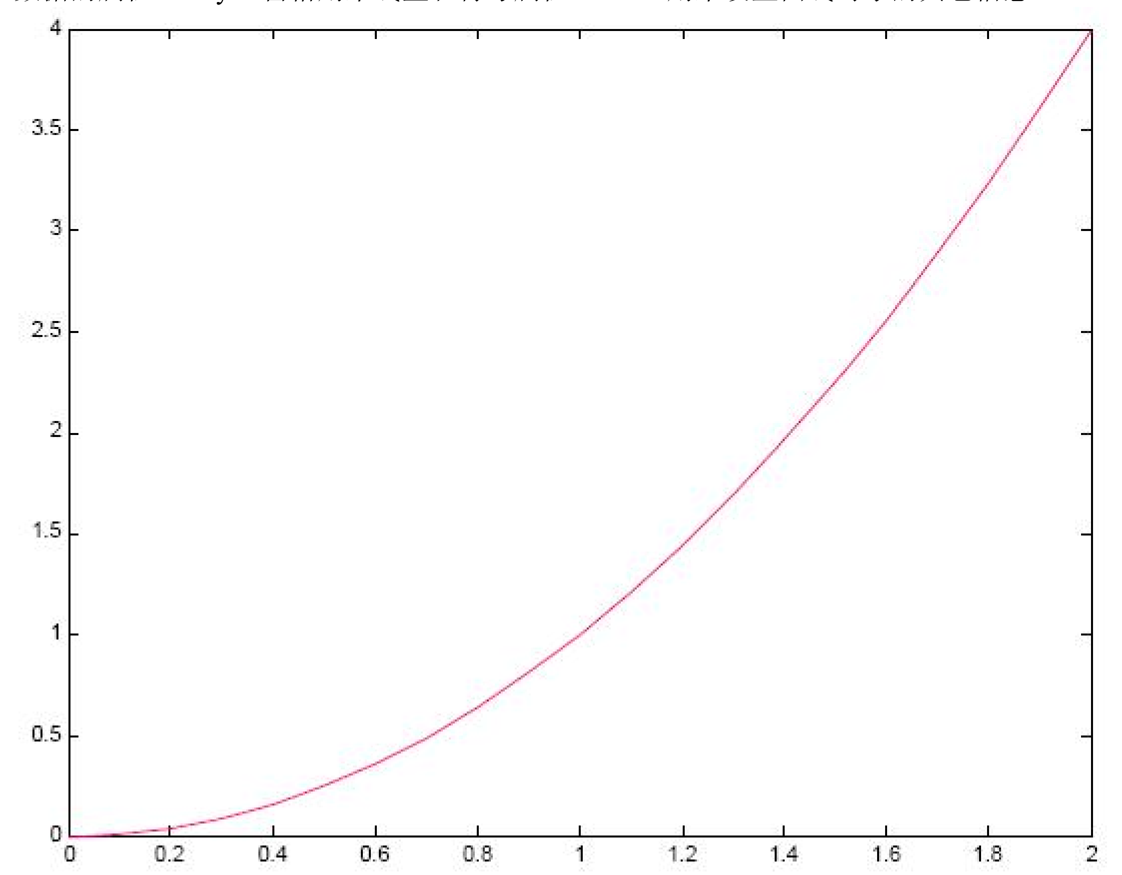
但对于最终的用户，他们要很容易地改变 **MATLAB** 对象的属性。属性编辑器是为了这个目的而设计的工具。启动属性编辑器的命令为

```
propedit(HandleList);
propedit;
```

这个函数第一个形式用于编辑所列出的句柄的属性，而这个函数的第二种形式用于编辑当前图象的属性。例如下面的语句创建函数 y(x)=在(0，2)x2 中的图象并打开属性编辑器，让用户间接地改变曲线的属性。

```
figure(2);
x = 0:0.1:2;
y = x .^2;
Hnd1 = plot(x, y);
propedit(Hnd1);
```

我们用这些语句调用属性编辑器，如图 9.3。属性编辑器包含了许多的窗格，用户可以根据对象的类型改变对象的属性。在例子中讨论的曲线对象，它窗格包括"Date"，"Style"，和"Info"。"Date"窗格允许用户选择和修改所要显示的数据它可以修改 X 数据，Y 数据和 Z 数据的属性。"Style"窗格用来线型和符号属性，"Info"用来设置曲线对象的其它信息。



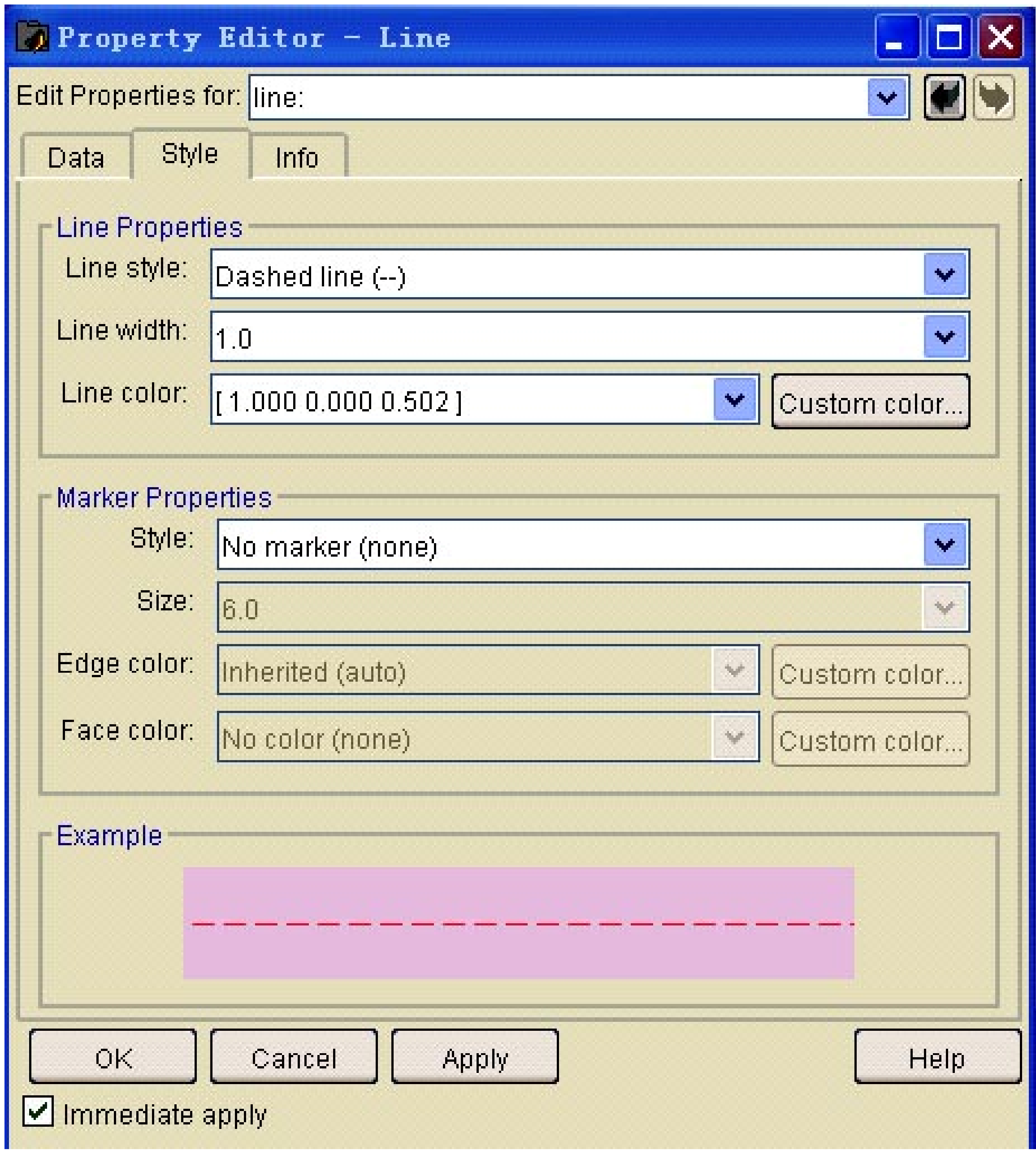


图 9.3 编辑曲线对象的属性编辑器。

属性编辑器可以用图象工具条上的按钮调用，然后双击你所要编辑的对象。

例 9.1

底层图形命令的应用

函数 sinc(x)的定义如下

$$\operatorname{sinc} x = \begin{cases} \dfrac{\sin x}{x} & x \neq 0 \\ 1 & x = 0 \end{cases} \tag{9.1}$$

x 的取值从-3π 到 3π，画出这个函数的图象，用句柄图形函数画出图象，满足下面的要求

1.使图象背景为粉红色

2.只在 y 轴上有网格线

3.曲线为 3point 宽，桔黄色的实线

答案

为了创建图象，我们需要计算 x 从-3π 到 3π 之间的函数 sincx，然后用 plot 命令画出它的图象。plot 命令这条直线的句柄。

画完直线后，我们需要修改 figure 对象的颜色和 axes 对象的网格状态，以及 line 对象的颜色与线宽。这些修改需要我们访问图对象，axes 对象，line 对象的句柄。图对象的句柄由函数 gcf 返回，axes 对象的句柄由函数 gca 返回，line 对象由 plot 函数返回。

需要修改的底层图形属性可以在 **MATLAB** 在线帮助工作台文件中找到，在主题"Handle Graphics Objects"目录下。它们包括当前图象的"color"属性，当前的坐标系的"YGrid"属性，以及曲线的"LineWidth"属性和"color"属性。

1.陈述问题

画出函数 sincx 的图象，x 的取值从-3π 到 3π，图象背景为粉红色，只在 y 轴上有网格线，曲线为 3point 宽，桔黄色的实线

2.定义的输入与输出

这个程序无输入，输出为指定的图象。

3.设计算法

这个问题可分为三大步

```
Calculate sinc(x)
Plot sinc(x)
Modify the required graphics object properties
```

这个程序的第一大步是计算 x 从-3π 到 3π 之间的函数 sincx。这个工作可以用向量化语句完成，但向量化语句在 x=0 时会产生一个 NaN，因为 0/0 没有意义。在画这个函数之前我们必须用 1.0 替换 NaN。这个步骤的伪代码为

```
% Calculate sinc(x)
x = -3*pi:pi/10:3*pi
y = sin(x) ./ x
%Find the zero value and fix it up. The zero is
%located in the middle of the x array.
index = fix(length(y)/2) + 1
y(index) = 1
```

下一步，我们必须画出这个函数的图象，并把所要修改的曲线的句柄存入一个变量。这个步骤的伪代码为

```
Hndl = plot(x, y);
```

现在，我们必须用句柄图形修改图象背景，y 轴上的网格线，线宽和线色。图对象的句柄由函数 gcf 返回，axes 对象的句柄由函数 gca 返回，line 对象由 plot 函数返回。

粉红色背景可由 RGB 向量[1 0.8 0.8]创建，桔黄色曲线可以由 RGB 向量[1 0.5 0]创建。

伪代码如下

```
set(gcf, 'Color', [1 0.8 0.8])
set(gca, 'YGrid', 'on')
set(Hndl, 'Color', [1 0.5 0], 'LineWidth', 3)
```

4.把算法转化为 **MATLAB** 语言

```
% Script file: plotsinc.m
%
% Purpose:
% This program illustrates the use of handle graphics
% commands by creating a plot of sinc(x) from -3*pi to
% 3*pi, and modifying the characteristics of the figure,
% axes, and line using the "set" function.
%
% Record of revisions:
% Date     Programmer     Description of change
% ====     ==========     =====================
% 11/22/97 S. J. Chapman Original code
```

```
%
% Define variables:
% Hndl -- Handle of line
% x -- Independent variable
% y -- sinc(x)
% Calculate sinc(x)
x = -3*pi:pi/10:3*pi;
y = sin(x) ./ x;
% Find the zero value and fix it up. The zero is
% located in the middle of the x array.
index = fix(length(y)/2) + 1;
y(index) = 1;
% Plot the function.
Hndl = plot(x,y);
% Now modify the figure to create a pink background,
% modify the axis to turn on y-axis grid lines, and
% modify the line to be a 2-point wide orange line.
set(gcf,'Color',[1 0.8 0.8]);
set(gca,'YGrid','on');
set(Hndl,'Color',[1 0.5 0],'LineWidth',3);
```

5.检测程序

这个程序的检测是非常简单的，我们只要运行这个程序，并检查产生的图象就可以了。产生的图象如图 9.4 所示，它就是我们需要的样式。

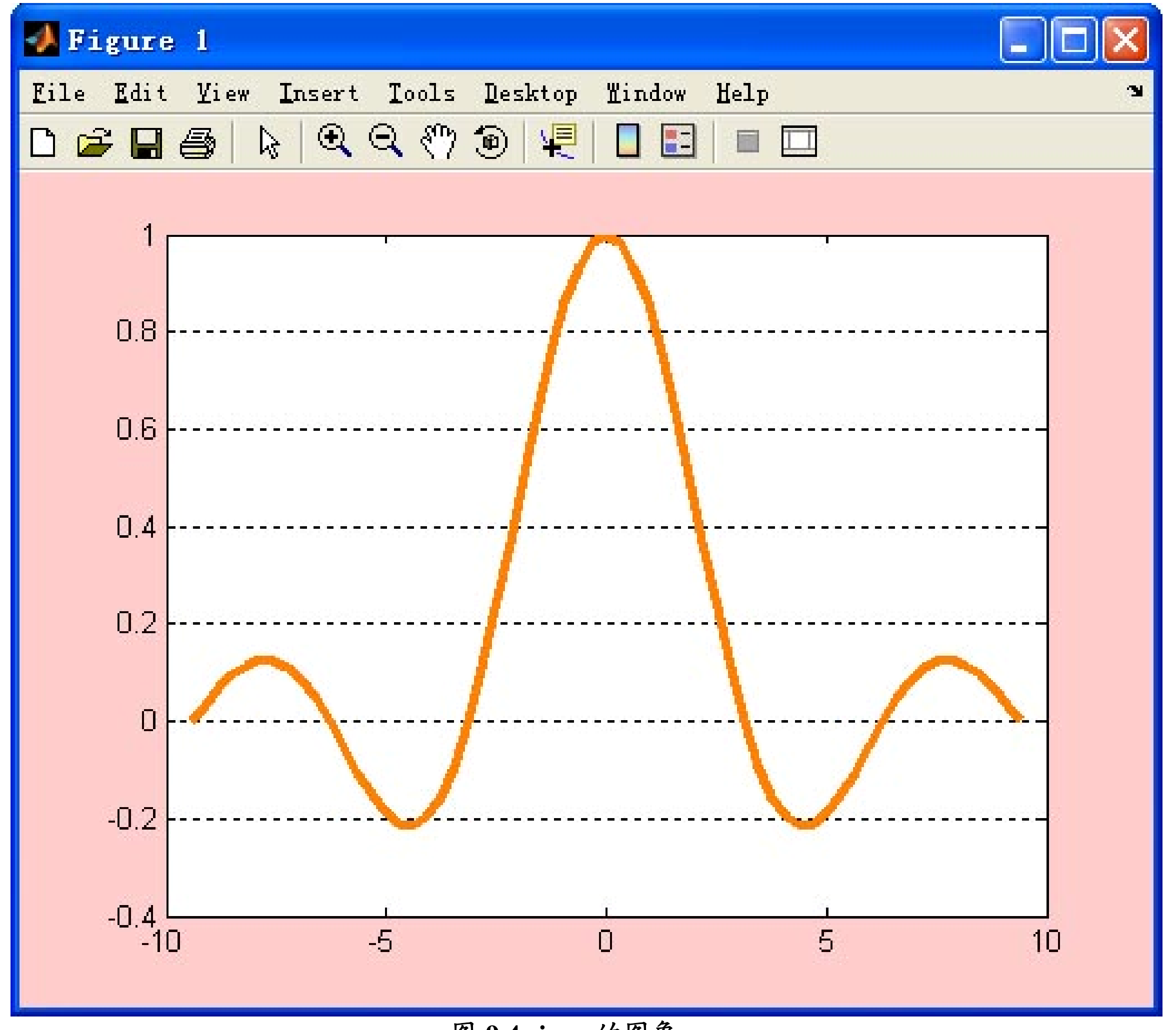


图 9.4 sincx 的图象

9.4 用 set 函数列出可能属性值

函数用于提供所有可能的属性值列表。如果在调用函数 set 时，只包括属性名而不包括相应的属性值，那么函数 set 就会返回所有的合法属性值。例如，命令 set(Hnd1，"LineStyle") 将返回所有可能的线型，大括号中是默认的线型。

```
>> set(Hndl,'LineStyle')
[ {-} | -- | : | -. | none ]
```

这个函数的合法包括和"none"，第一个是默认的类型。

```
>> set(Hndl,'LineWidth')
A line's "LineWidth" property does not have a fixed set of property values.
```

函数 set(Hnd1)返回一个对象的所有属性的所有可能的属性值。

```
>> set(Hndl)

ans =

              Color: {}
          EraseMode: {4x1 cell}
          LineStyle: {5x1 cell}
          LineWidth: {}
             Marker: {14x1 cell}
         MarkerSize: {}
    MarkerEdgeColor: {2x1 cell}
    MarkerFaceColor: {2x1 cell}
              XData: {}
              YData: {}
              ZData: {}
      ButtonDownFcn: {}
           Children: {}
           Clipping: {2x1 cell}
          CreateFcn: {}
          DeleteFcn: {}
         BusyAction: {2x1 cell}
   HandleVisibility: {3x1 cell}
            HitTest: {2x1 cell}
      Interruptible: {2x1 cell}
           Selected: {2x1 cell}
 SelectionHighlight: {2x1 cell}
                Tag: {}
      UIContextMenu: {}
           UserData: {}
            Visible: {2x1 cell}
             Parent: {}
        DisplayName: {}
          XDataMode: {2x1 cell}
        XDataSource: {}
        YDataSource: {}
        ZDataSource: {}
```

9.5 自定义数据

除了一个 GUI 对象定义的标准属性以外，程序可以定义所要控制的数据的特殊属性。程序员可以用附加属性把任意类型的数据添加到 GUI 对象中。任意数量的数据可以被存储，

并应用于各种目的。

自定义数据可以用近似标准属性的形式存储。每一个数据条目都有一个名字和值。数据变量可以用函数 setappdate 存储在一个对象，并用函数 getappdata 接收。

setappdate 函数的基本形式如下

```
setappdata(Hndl, 'DataName', DataValue);
```

其中 Hndl 是数据存入的对象的句柄，"DateName"是这个数据的名字，而 DateValue 是赋于是这个名字的值。注意数据值可以是数字，也可以是字符串。

例如，假设我们要定义两个特殊的数据值，其中一个用于存储发在指定图象中的错误数，另一个是用于描述最后发现的错误的字符串。这两个数据值的名字是"ErrorCount"和"LastError"。我们假设 H1 为这个图象的句柄，创建这些数据条目和初始化的命令为

```
setappdata(Hl,'ErrorCount',0);
setappdata(H1,'LastError','No error');
```

我们可以用 getappdata 函数随时调用这些数据。getappdata 的两种形式如下

```
value = getappdata(Hndl, 'DataName');
struct = getappdata(Hndl);
```

其中，Hnd1 是包含有这个数据的对象句柄，"DateName"是要调用的数据的名字，如果一个"DateName"被指定，那么与"DateName"相关的值就会被返回。如果没有被指定，那么所有与这个对象形字相关的自定义值就会以结构的形式被返回。数据条目名就是结构元素名。

对上面的例子来说，getappdata 将会产生下面的结果

```
>> value = getappdata(Hl, 'ErrorCount')
value =
        0
>> value = getappdata(Hl);
struct =
      ErrorCount:   0
      LastError:   'No error'
```

与自定义数据相关的函数被总结在表 9.1 中。

表 9.1 与自定义数据相关的函数

函数	描述
setappdata(Hndl, 'DataName', DataValue)	把 DataValue 存储在对象中的'DataName'，这个对象以 Hndl 为句柄。
value = getappdata(Hndl, 'DataName') struct = getappdata(Hndl)	从以 Hndl 句柄的对象重新调用程序，第一种形式只读取'DataName'中的数据，第二种形式重新所有的自定义数据。
isappdata(Hndl, 'DataName')	如果'DataName'在以 Hndl 为句柄的对象中有定义，那就会返回 1，否则返回 0。
isappdata(Hndl, 'DataName')	删除'DataName'，'DataName'是在以 Hndl 为句柄的对象中的自定义数据。

9.6 对象查找

每一个新的图象在从创建开始时就有它们自己的句柄，句柄可以由创建函数返回。

好的编程习惯

如果你打算修改你创建的对象的属性，那么请保存对象的句柄，为以后调用函数 get 和 set 做准备。

但是我们有时不能访问句柄。假设我们由于一些原因，丢失了对象的句柄。我们如何检测和图形对象呢?**MATLAB** 提供了四个专门的函数，用来帮助寻找对象的句柄。

- gcf　　返回当前图象的句柄
- gca　　返回当前图象中当前坐标系的句柄
- gco　　返回当前对象的句柄
- findobj　寻找指定属性值的图形对象

函数 gcf 返回当前图象的句柄。如果这个图象不存在，gcf 将会创建一个，并返回它的句柄。函数 gca 返回当前图象中当前坐标系的句柄，如果图象不存在，或当前图象中无坐标系，那么函数 gca 将创建一个坐标系，并返回它的句柄。函数 gco 的形式如下

```
H_obj = gco;
H_obj = gco(H_fig);
```

其中，H_obj 是一个对象的句柄，H_fig 是一个图象的句柄。这个函数的第一种形式返回当前图象中的当前对象的句柄。它的第二种形式返回一指定图象中的当前对象的句柄。

当前对象是指用鼠标单击的最一个对象。这个对象可以是除了根对象的任意图形对象。直到鼠标在图象内发生了单击事件，在图象内才有一个当前对象。在单击事件发生之后，函数 gco 将返回一个空数组[]，不像函数 gcf 和 gca，gco 如果不存在就自动创建。

一旦我们得知了一个对象的句柄，我们可以通过检测"Type"属性去时来确定对象的类型。"Type"属性是一个字符串，例如"图"，"line"，"text"等等。

```
H_obj = gco;
type = get(H_obj, 'Type')
```

查找任意一个 **MATLAB** 对象最简单的方法是用 findobj 函数。它的基本形式如下

```
Hndls = findobj('PropertyName1',value1, ...)
```

这个命令起始于根对象，并搜索所有的对象，找出含有指定属性，指定值的对象。注意可以指定多个属性/值，findobj 只返回与之匹配的对象句柄。

例如，假设我们已经创建了图 1 和图 3。那么函数 findobj("Type"，"图")将会返回结果

```
>> H_fig = findobj('Type', 'figure')
H_fig =
     3
     1
```

函数 findobj 的这种形式非常的有用，但却比较慢，因为它必须对整个对象树进行搜索。如果你必须多次用到一对象，只调用一次函数 findobj，为了重复利用句柄，句柄应存储下来。

限定搜索对象的数目能够加快函数运行的速度。它的形式为

```
Hndls = findobj(SrchHndls, 'PropertyName1', value1, ...)
```

在这里，只有数组 srchHndls 和它的子数组中的句柄，才在搜索的范围内。例如你想找到图 1 中的虚线。它的命令为

```
Hndls = findobj(1, 'Type', 'line', 'LineStyle', '--');
```

好的编程习惯
如果有可能的话，限定函数 findobj 的搜索范围将能加快函数的运行速度。

9.7 用鼠标选择对象

函数 gco 将返回当前对象，当前对象是指用鼠标最后一次单击的对象。每一个对象都有一个与之相关的**可选择区**，在可选择区内任意一个单击都可以看作对这个对象的单击。对于细小的对象(例如线，点)来说，这种特性是非常重要的。可选择区的宽度和形状由对象的类型确定。例如，一个曲线的可选择区在离直线 5pixel 的范围内，而一个表面，一个小块和文本对象的可选择区是包含这些对象的最小长方形。

对于一个坐标系对象来说，它的可选择区是坐标轴区域加上标题和标签的区域。但是在坐标轴内的曲线对象或其他对象有更高的优先权，你必须在单击坐标内的一点，并且不靠近直线和文本。如果单击坐标外的图象将会选择图象本身。

如果一个用户单击了两个或多个对象的所在点，例如两线的交插点将会有什么事情发生。这取决于每一个对象**堆垛顺序(stacking order)**。堆垛顺序是 **MATLAB** 选择对象的顺序。在一个图象中所有的"子对象"属性句柄顺序就是堆垛顺序。如果单击了两个或多个对象的所在点，在堆垛顺序的优先权高的将会被选择。

当选择图形对象时，我们有时可以调用 **MATLAB** 内建函数 waitforbuttonpress。这个函数的形式为

```
k = waitforbuttonpress
```

当这个函数运行时，它将会暂停程序，直到任意键按下或鼠标单击事件发生后，程序才恢复运行。如果按下了鼠标键函数将会返回 0，按下任意键，函数将会 1。

函数经常用于暂停程序。当鼠标单击事件发生后，程序将会用 gco 函数恢复选择对象的句柄。

例 9.2

图形对象的选择

在本例中的程序可以探测图形对象的属性，并显示如何用函数 waitforbuttonpress 和 gco 选择对象。程序允许用户可以多次重复选择对象。

```
% Script file: select_object.m
%
% Purpose:
% This program illustrates the use of waitforbuttonpress
% and gco to select graphics objects. It creates a plot
% of sin(x) and cos(x), and then allows a user to select
% any object and examine its properties. The program
% terminates when a key press occurs.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== =====================
% 11/23/97 S. J. Chapman Original code
%
% Define variables:
% details -- Object details
% H1 -- Handle of sine line
% H2 -- Handle of cosine line
% Handle -- Handle of current object
% k -- Result of waitforbuttonpress
% type -- Object type
% x -- Independent variable
% y1 -- sin(x)
% y2 -- cos(x)
% yn -- Yes/No
% Calculate sin(x) and cos(x)
x = -3*pi:pi/10:3*pi;
y1 = sin(x);
y2 = cos(x);
% Plot the functions.
H1 = plot(x,y1);
set(H1,'LineWidth',2);
hold on;
```

```
H2 = plot(x,y2);
set(H2,'LineWidth',2,'LineStyle',':','Color','r');
title('\bfPlot of sin \itx \rm\bf and cos \itx');
xlabel('\bf\itx');
ylabel('\bfsin \itx \rm\bf and cos \itx');
legend('sine','cosine');
hold off;
% Now set up a loop and wait for a mouse click.
k = waitforbuttonpress;
while k == 0
    % Get the handle of the object
    Handle = gco;
    % Get the type of this object.
    type = get(Handle,'Type');
    % Display object type
    disp (['Object type = ' type '.']);
    % Do we display the details?
    yn = input('Do you want to display details? (y/n) ','s');
    if yn == 'y'
        details = get(Handle);
        disp(details);
    end
    % Check for another mouse click
    k = waitforbuttonpress;
end
```

程序运行后，得到的结果如图 9.5 所示

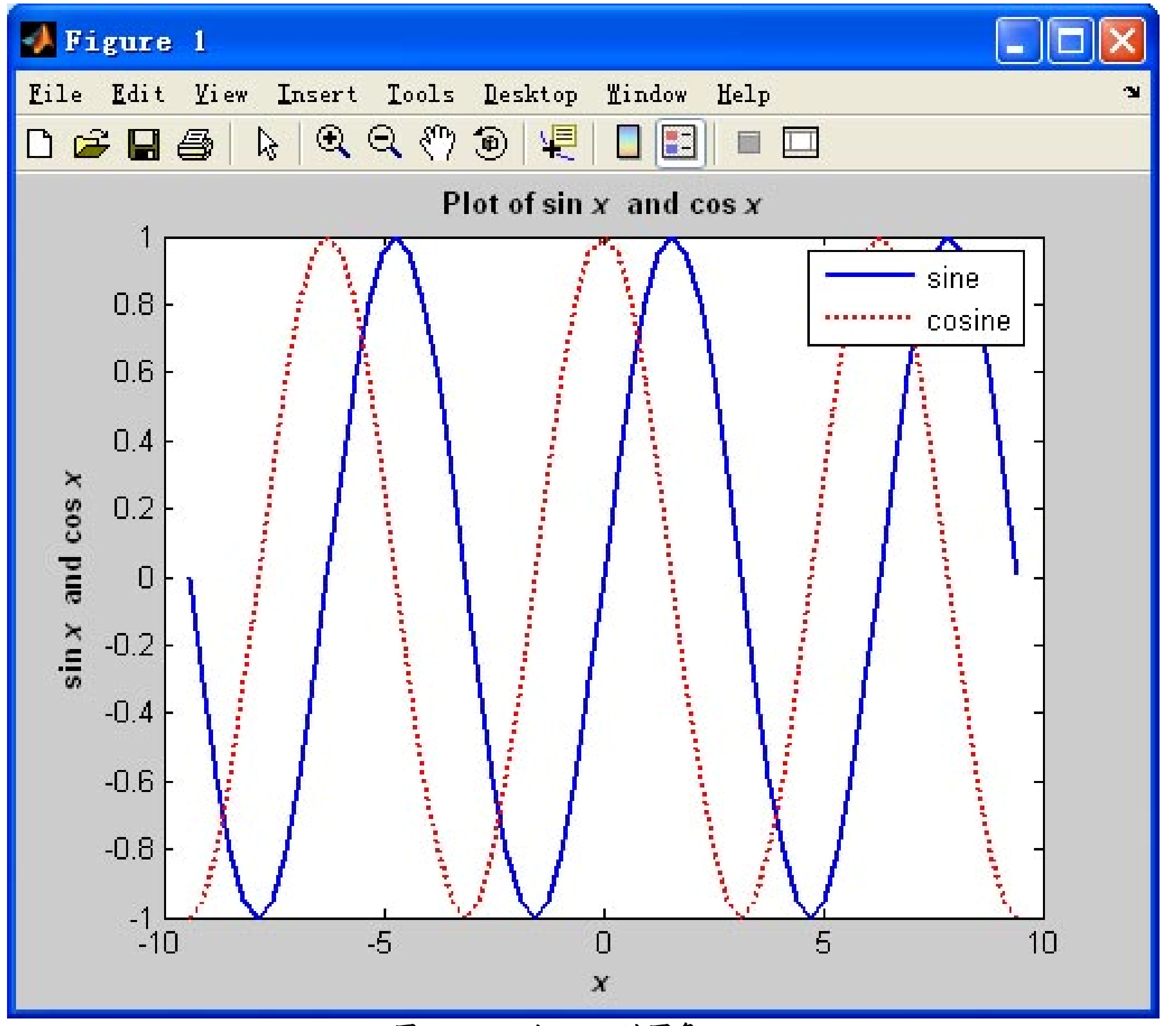


图 9.5 sinx 和 cosx 的图象。

我们可以在图象内单击各种对象，并查看它的类型。

9.8 位置和单位

许多的 **MATLAB** 对象都包括位置("position")属性，它用来指定对象在计算机屏幕的位置和大小。这个属性在不同类型的对象中有细节的差别，这一点将在本节中描述。

9.8.1 图象(figure)对象的位置

一个图象(图)的位置("position")用一个 4 元素行向量指定在计算机屏幕内的位置。在这个向量中的值为[left bottom width height]，其中 left 是指图象的左边界，bottom 是指图象的底边界，width 是指图象的宽度，height 是指图象的高度。它的这些位置值的单位可以用对象的"Units"属性指定。例如，与当前图象的位置和单位可以用下面的语句得到。

```
>> get(gcf,'Position')
ans =
    128    259    506    373
>> get(gcf,'Units')
ans =
```

```
pixels
```

这些信息说明当前图象窗口的左下角距屏幕右边的距离为 176pixel，距屏幕底边的距离为 204pixel。，图象的宽度为 672pixel，上下高度为 504pixel。注意这是图象的可作图区，包括边界，滚动条，菜单，还有图象的标题区。

单位("units")属性的默认值为像素(pixels)，但是它的属性值还可以为英尺(inches)，公分(centimeters)，点(points)，或归一化坐标(normalixed coordinates)。像素代表了屏幕像素，即在屏幕上可表示出来的最小的对象。典型的计算机屏幕最小分辨为 640×480，在屏幕的每一个位置都有超过 1000 的像素。因为像素数因机算机屏幕的不同而不同，所以指定对象的大小也会随之改变。

归一化坐标是在 0 到 1 范围内。在归一化坐标中，屏幕的左下角为[0,0]右上角为[1.0, 1.0]。

如果对象的位置归一化坐标系的形式描述，那么不同分辨率的显示器上对象的相对位置是固定的。例如，下面的语句创建了一个图象，把图象放置在屏幕的上部，而不用考虑显示器的大小。

```
H = figure(1)
set(H,'units', 'normalized','position',[0 .5 .5 .45])
```

好的编程习惯

如果你想把对象放置在窗口的特定位置，最好的方法是用归一化坐标，因为不用考虑显示器的大小。

9.8.2 坐标系对象和 uicontrol 对象的位置

坐标系对象和 uicontrol 对象的位置同样可以用一个 4 元素向量表示，但它是相对于 figure 对象的位置。一般说来，所有子对象的"position"属性都与它的父对象相关。

默认地，坐标系对象在一图象内的位置是有归一化单位指定的，(0，0)代表图象的左下角，(1，1)代表图象的右上角。

9.8.3 文本(text)对象的位置

与其他对象不同，文本(text)对象有一个位置属性，包含两个或三个元素。这些元素为坐标系对象中文本对象的 x，y 和 z 轴。注意都显示在坐标轴上。

放置在某一特定点的文本对象的位置可由这个对象的 HorizontalAlignment 和 VerticalAlignment 属性控制。HorizontalAlignment 的属性可以是{Left}。Center，或 Right。VerticalAlignment 的属性值可以为 Top，cap，{Middle}，Baseline 或 Bottom。

文本对象的大小由字体大小和字符数决定，所以没有高度和宽度值与之相连。

例 9.3

设置一个图象内对象的位置

正如我们前面所提到的，坐标系的位置与包含它的图象窗口的左下角有关，而文本对象的位置与坐标系的位置相关。

为了说明如何在一图象窗口中设置图形对象的位置，我们将编写一个程序，用它在单个的图象窗口内创建两个交迭的坐标系。

第一个坐标系将用来显示函数 sinx 的图象，并带有相关文本说明。第二个坐标系用来显示函数 cosx 的图象，并在坐标系的左下角有相关的文本说明。

用来创建图象的程序如下所示。注意我们用图函数来创建一个空图象，然后两个 axes 函数在图象窗口中创建两个坐标系。函数 axes 的位置可以用相对于图象窗口的归一化单位指定，所以第一个坐标系起始于(0.05,0.05)，位于图象窗口的左下角，第二坐标系起始于(0.45,0.45)，位于图象的右上角。每个坐标系都有合适的函数进行作图。

第一个坐标系中的文本对象的位置为(-π, 0)，它是曲线上的一点。当我们选择 HorizontalAlignment 的属性值为 right，那么点(-π, 0)则在文本字符串的右边。所以在最终的图象中，文本就会显示在位置点的左边(这对于新程序员来说很容易迷惑)。

在第二个坐标系中的文本对象的位置为(7.5, 0.9)，它位于坐标轴的左下方。这个字符串用 HorizontalAlignment 属性的默认值"left"，点(7.5，0.9)则在文本字符串的右边。所以在最终的图象中，文本就会显示在位置点的右边。

```
%Script file: position_object.m
%
%   Purpose:
%   This program illustrates the positioning of graphics
%   objects. It creates a figure and then places
%   two overlapping sets of axes on the figure. The first
%   set fo axes is placed in the lower left corner of
%   the figure. and contains a plot of sin(x), The second
%   set of axes is placed in the upper right corner of the
%   figure, and contains a plot of cos(x). Then two
%   text strings are added to the axes, illustrating the
%   positioning of text within axes.
%
%   Record of revisions:
%       Date        Programmer        Description fo change
%       =====       ==========        =====================
%   02/26/99        S.J.Chapman       Original code
%
%   Define varibles:
%   H1          --Handle of sine line
%   H2          --Handle of sosine line
%   Ha1         --Handle of first axes
%   Ha2         --Handle of second axes
%   x           --Independent variable
%   y1          --sin(x)
%   y2          --cos(x)
%   Calculate sin(x) and cos(x)
x = -2*pi:pi/10:2*pi;
y1 = sin(x);
y2 = cos(x);
%   Create a new figure
figure;
%   Create the first set of axes and plot sin(x).
%   Note that the position of the axes is expressed
%   in normalized units.
Ha1 = axes('Position',[.05 .05 .5 .5]);
H1 = plot(x, y1);
set(H1,'LineWidth',2);
title('\bfPlot of sin \itx');
xlabel('\bf\itx');
ylabel('\bfsin \itx');
axis([-8 8 -1 1]);

%   Create the second set of  axes and plot cos(x).
%   Note that the position of the axes is expressed
%   in  normalized units.
```

```
Ha2 = axes('Position',[.45 .45 .5 .5]);
H2 = plot(x, y1);
set(H2,'LineWidth',2,'Color','r','LineStyle','--');
title('\bfPlot of cos \itx');
xlabel('\bf\itx');
ylabel('\bfsin \itx');
axis([-8 8 -1 1]);

%      Create a text string attached to the line on the first
%      set of axes.
axes(Ha1);
text(-pi,0.0,'min(x)\rightarrow','HorizontalAlignment','right');

%      Create a text string in the lower left corner
%      of the second set of axes.
axes(Ha2);
text(-7.5,-0.9,'Text string 2');
```

当这个程序执行后，产生的图象如图 9.6 所示。你就应当在你的计算机上重复地执行这人程序，所要画的对象的大小与位置，观察结果。

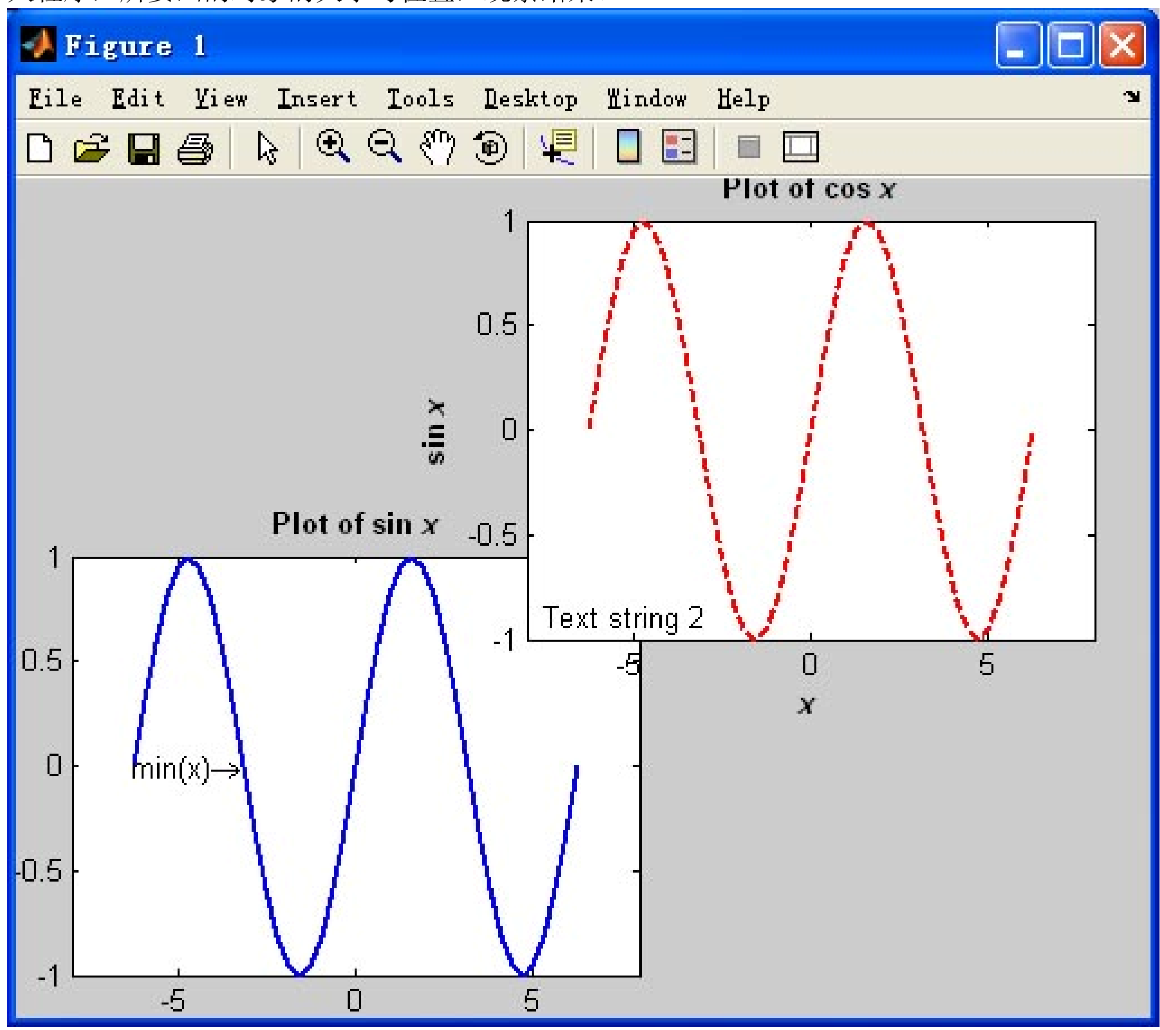


图 9.6 程序 position_object 的结果。

9.9 打印位置

属性"Position"和"Units"用来指定图象在计算机屏幕上的位置。还有其他的五个属性用于

指定图象在打印纸上的位置。这些属性被总结在表 9.2 中。

表 9.2 与打印相关的图象属性

参数	描述
PaperUnits	度量纸张的单位 [{inches} | centimeters | normalized | points]
PaperOrientation	[{portrait} | landscape]
PaperPosition	位置向量，形式为[left, bottom, width, height]，单位是 PaperUnits。
PaperSize	包含纸张大小两个元素的向量，例如[8.5 11]
PaperType	设置纸张的类型，注意设置这个属性会自动更新纸张的 PaperSize 属性。[{usletter} | uslegal | a3 | a4letter | a5 | b4| tabloid]

例如，我们用 landscape 模式，用归一化单位在 A4 纸上打印一个图象。我们可以设置下面的属性。

```
set(Hndl, 'PaperType', 'a4letter')
set(Hndl, 'PaperOrientation', 'landscape')
set(\Hndl, 'PaperUnits', 'normalized');
```

9.10 默认和 factory 属性

当一个对象被创建时，**MATLAB** 就会把默认的属性值赋值于每一个对象。如果这些属性值不是你想要的，那么你必须用 set 函数选择你想要的值。如果你想更改你创建的每一个对象的一个属性，这个过程将变得非常麻烦。由于这个原因，**MATLAB** 允许你修改默认值本身，所以当他们被创建时，所有的对象都会继承所有正确的属性值。

当一个图形对象被创建时，**MATLAB** 就会通过检测对象的父对象来寻找每一个属性的默认值。如果父对象设置了默认值，那么这个值就会被应用。如果没有设置默认值，那么 **MATLAB** 就会检测父对象的父对象，看是否有默认值。以此类推，直到根对象。在这个过程中，**MATLAB** 会应用第一次遇到的默认值。

默认属性可以在优先级高的图形对象中的任意一点设置。例如，默认的图的颜色在根对象中设置，而在这之后的所有图象都有一个新的默认颜色。从另一方面说，默认的坐标轴颜色可以在根对象或图象对象设置。如果坐标的默认颜色在根目录中设置，那么它将应用于所有图象的所有新坐标轴，如果默认的坐标轴颜色在图象对象中设置，它将在当前图象窗中的新坐标轴。

默认值的设置要用一个字符串，这个字符串由"Default"，对象类型和属性名组成。所以默认图象颜色可以通过属性"DefaultFigureColor"来设置，默认的坐标轴颜色可以通过属性"DefaultAxesColor"设置。下面是设置默认值的一些例子

```
set(0, 'DefaultFigureColor', 'y')      黄色图象背景
set(0, 'DefaultAxesColor', 'r')        红色坐标系背景——所有图象中的坐标轴
set(gcf, 'DefaultAxesColor', 'r')      红色坐标系背景——当前图象坐标轴
set(gca, 'DefaultLineStyle', ':')      只在当前坐标系中设置默认线型为虚线
```

如果你要对已存在的对象的属性进行修改，那么在用完这个属性之后，最好恢复原来的条件。如果你在一个函数中修改了一个对象的默认属性值，保存它原来的值，并在跳出这个函数之前恢复它们。例如，假充我们用归一化单位创建一系列的图象，我们可以用下面的保存和修复原来的单位。

```
saveunits = get(0, 'DefaultFigureUnits');
set(0, 'DefaultFigureUnits', 'normalized');
...
<MATLAB statements>
...
set(0, 'DefaultFigureUnits', saveunits);
```

如果你想要定制 **MATLAB**，每一次都有不同的默认值，那么每次当 **MATLAB** 启动时你必须对根对象设置默认值。最简单的方法是把默认值存入 startup.m 文件，每次 **MATLAB** 启动时都会自动执行。例如，假设你经常使用 A4 纸，并在图象中经常加入网格线。那么你可以把下面的语句加入到 startup.m 文件中。

```
set(0, 'DefaultFigurePaperType', 'a4letter');
set(0, 'DefaultAxesXGrid', 'on');
set(0, 'DefaultAxesYGrid', 'on');
set(0, 'DefaultAxesZGrid', 'on');
```

有三种特殊值字符串用于句柄图形:"remove"，"factory"和"default"。如果你已经为一个属性设置了默认值，那么"remove"值将会删除你所设置的默认值。例如，假设你设置默认的图象颜色为黄色。

```
set(0, 'DefaultFigureColor', 'y');
```

调用下面的函数将会取消当前的默认值并恢复先前的默认值。

```
set(0, 'DefaultFigureColor', 'remove');
```

字符串"factory"允许临时跳过当前的默认值，并使用原来的 **MATLAB** 的默认值。例如，尽管当前的默认颜色为黄色，下面的语句将会用 factory 创建下面的图象。

```
set(0, 'DefaultFigureColor', 'y');
figure('Color', 'factory');
```

第三个特殊的属性值字符串是 default，这个属性值迫使 **MATLAB** 搜索对象层次结构，直到查到所需属性的一个默认值。如果找到，它就使用该默认值。如果查到根对象，没有找到用户定义的默认值，**MATLAB** 就使用 factory 默认值。它的应用说明如下

```
%    Set default values
set(0, 'DefaultLineColor', 'k');       %    root default = black
set(gcf, 'DefaultLineColor', 'g');     %    figure default = green
%    Create a line on the current axes. This line is green.
Hndl = plot(randn(1, 10));
set(Hndl, 'Color', 'default');
pause(2);
%    Now clear the figure default and set the line color to the new
%    default. The line is now black.
set(gcf, 'DefaultLineColor', 'remove');
set(Hndl, 'Color', 'default');
```

9.11 图形对象属性

由于有成百上千的图形对象属性，我们在这里不一一讨论了。我们可以通过 **MATLAB** 帮助台得到所有属性。

9.12 总结

好的编程方法总结

1. 如果你打算修改你创建的对象的属性，那么请保存对象的句柄，为以后调用函数

get 和 set 做准备。

2. 如果有可能的话，限定函数 findobj 的搜索范围将能加快函数的运行速度。

3. 如果你想把对象放置在窗口的特定位置，最好的方法是用归一化坐标，因为不用考虑显示器的大小。

MATLAB 总结

- gcf　返回当前图象的句柄
- gca　返回当前图象中当前坐标系的句柄
- gco　返回当前对象的句柄
- findobj　寻找指定属性值的图形对象

9.13 练习

9.1 句柄语句什么意思?**MATLAB** 图形对象的优先级是怎样的?

9.2 用 **MATLAB** 帮助工作台了解图对象的 Name 和 Number Title 属性。画出函数 $y(x) = e^x(-2\leq x\leq 2)$的图象。改变上面提到的图象属性。禁止更改图象数，禁止增加"plot window"的标题。

9.3 编写一个程序，修改图象的默认颜色为桔黄色，默认线宽为 3.0point。那么用下面的等式创建图象

$$x(t) = 10\cos t$$
$$y(t) = 6\sin t$$

其中 t 的取值从 $t = 0$ 到 $t = 2\pi$。

9.4 用 **MATLAB** 帮助工作台了解坐标系对象的 CurrentPoint 属性。用这个属性编写一个程序。创建一个坐标系对象，并在坐标内画出鼠标单击过的点，然后用直线相连。用函数 waitforbuttonpress 等待鼠标单击，并在每一次单击后更新图象。当键盘事件发生后，中止程序。

9.5 用 **MATLAB** 程序画出下面函数的图象

$$x(t) = \cos\frac{t}{\pi}$$

$$x(t) = 2\sin\frac{t}{2\pi}$$

其中$-2\leqslant t\leqslant 2$。这个程序应当等待鼠标单击，如果鼠标单击在两条直线上的一条，程序就会随机地改变这条直线的颜色。用函数 waitforbuttonpress 等待鼠标单击，并在每一次单击后更新图象。用函数 gco 判断单击的对象，并用 Type 属性判断单击的对象是否为直线。

9.6 创建一个 **MATLAB** 图象，并把一些自定义数据变量存储在图象中。然后用 file/save 把图象存储到图象文件中(*.fig)。重启 **MATLAB** 程序，加载图象文件，并用函数 getappdata 查看自定义数据。这些数据被存储了吗?

第十章 用户图形界面

用户图形界面（GUI）是程序的图形化界面。一个好的 GUI 能够使程序更加容易的使用。它提供用户一个常见的界面，还提供一些控件，例如，按钮，列表框，滑块，菜单等。用户图形界面应当是易理解且操作是可以预告的，所以当用户进行某一项操作，它知道如何去做。例如，当鼠标在一个按钮上发生了单击事件，用户图形界面初始化它的操作，并在按钮的标签上对这个操作进行描述。

本章将向大家 **MATLAB** 用户图形界面的基本元素。本章不会对部件和 GUI 特性进行全部的描述，但是它将为你的程序提供必须的 GUI 元素。

10.1 用户界形界面是如何工作的

用户图形界为用户提供了一个熟悉的工作环境。这个环境包括按钮，列表框，菜单，文本框等等，所有的这些控件对用户来说非常地熟悉。所以能够应用它操作应用程序，而不用直接调用操作函数。但是，对于程序员来说，GUI 比较难的，因为程序的每一个控件都必须为鼠标单击做好准备。像鼠标单击这样的输入就是我们熟知的**事件**，而对事件有反应的程序，我们称之为**事件驱动**。

创建 **MATLAB** 用户图形界面必须由三个基本元素：

1.组件　　在 **MATLAB** GUI 中的每一个项目(按钮，标签，编辑框等)都是一个图形化组件组件可分为三类：**图形化控件**(按钮，编辑框，列表，滑动条等)，**静态元素**(窗口和文本字符串)，**菜单和坐标系**，图形化控件和静态元素由函数 uicontrol 创建，菜单由函数 uimenu 和 uicontextmenu 创建，坐标系经常用于显示图形化数据，由函数 axes 创建。

2.图象窗口 GUI 的每一个组件都必须安排图象窗口中。以前，我们在画数据图象时，图象窗口会被自动创建。但我们还可以用函数 figure 来创建空图象窗口，空图象窗口经常用于放置各种类型的组件。

3.响应　　最后，如果用户用鼠标单击或用键盘键入一些信息，那么程序就要有相应的动作。鼠标单击或键入信息是一个事件，如果 **MATLAB** 程序运行相应的函数，那么 **MATLAB** 函数肯定会有所反应。例如，如果用户单击一按钮，这个事件必然导致相应的 **MATLAB** 语句执行。这些相应的语句被称为响应。只要执行 GUI 的单个图形组件，必须有一个响应。

基本的 GUI 元素被总结在表 10.1 中，一些元素的例子被显示在图 10.中。我们将会学习这些例子，并通过它们说明 GUI 是如何工作的。

10.2 创建并显示用记图形界面

我们用工具 guide 来创建 **MATLAB** 用户图形界面，guide 是 GUI 集成开发环境。此工具允许程序员安排用读图形界面，选择和放置所需的 GUI 组件。一旦这些组件放置成功，程序员就能够编辑它们的属性：名字，颜色，大小，字体，所要显示的文本等等。当 guide 保存了这个用户图形界面之后它将会自动创建一个包括有骨干函数的工作程序，程序员可以利用这些程序执行用户图形界面的执行动作。

当执行 guide 语句时，**MATLAB** 将会创建一个版面编辑器（layout editor），如图图 10.2 所示。带有网格线的大空白区域被称之为布局区（the layout area）。用户可以通过单击你所需要的组件创建任意的目的 **MATLAB** 组件，然后通过拖动它的轮廓线，把它放置在布局区

内。在这个窗口的上部用一个带有许多有用工具的工具条，它允许用户分配和联合 GUI 组件，修改每一个 GUI 组件的属性，在用户图形界面中添加菜单等。

创建一个 **MATLAB** 用户图形界面的基本步骤为：

1. 决定这个用户图形界面需要什么样的元素，每个元素需要什么样的函数。在纸上手动粗略地画出组件的布局图。

表 10.1 一些基本的 GUI 组件

元素	创建元素的函数	描述
图形控件		
按钮 (pushbutton)	uicontrol	单击它将会产生一个响应
开关按钮 (togglebutton)	uicontrol	开关按钮有两种状态“on”，“off”，每单击一次，改变一次状态。每一个单击一次产生一个响应。
单选按钮 (radiobutton)	uicontrol	当单选按钮处于 on 状态，则圆圈中有一个点
复选按钮 (checkbox)	uicontrol	当复选按钮处于 on 状态时，复选按钮中有一个对号
文本编辑框 (editbox)	uicontrol	编辑框用于显示文本字符串，并允许用户修改所要显示的信息。当按下回车键后将产生响应
列表框 (listbox)	uicontrol	列表框可显示一系文本字符串，用于可用单击或双击选择其中的一个字符串。当用户选择了其中一个字符串后，它将会有一个响应。
下拉菜单（popup Menus）	uicontrol	下拉菜单用于显示一系列的文本字符串，当单击时将会产生响应。当下拉菜单没有点击时，只有当前选择的字符串可见
滑动条（slider）	uicontrol	每改变一次滑动条都会有一次响应。
静态元素		
框架（frame）	uicontrol	框架是一个长方形，用于联合其他控件。而它则不会产生反应
文本域（textfield）	uicontrol	标签是在图像窗口内某一点上的字符串。
菜单和坐标系		
菜单项（menuitems）	Uimenu	创建一个菜单项。当鼠标在它们上单击时，它将会产生一个响应
右键菜单（contextmenus）	Uicontextmenu	创建一个右键菜单
坐标系（axes）	Axes	用来创建一个新的坐标系。

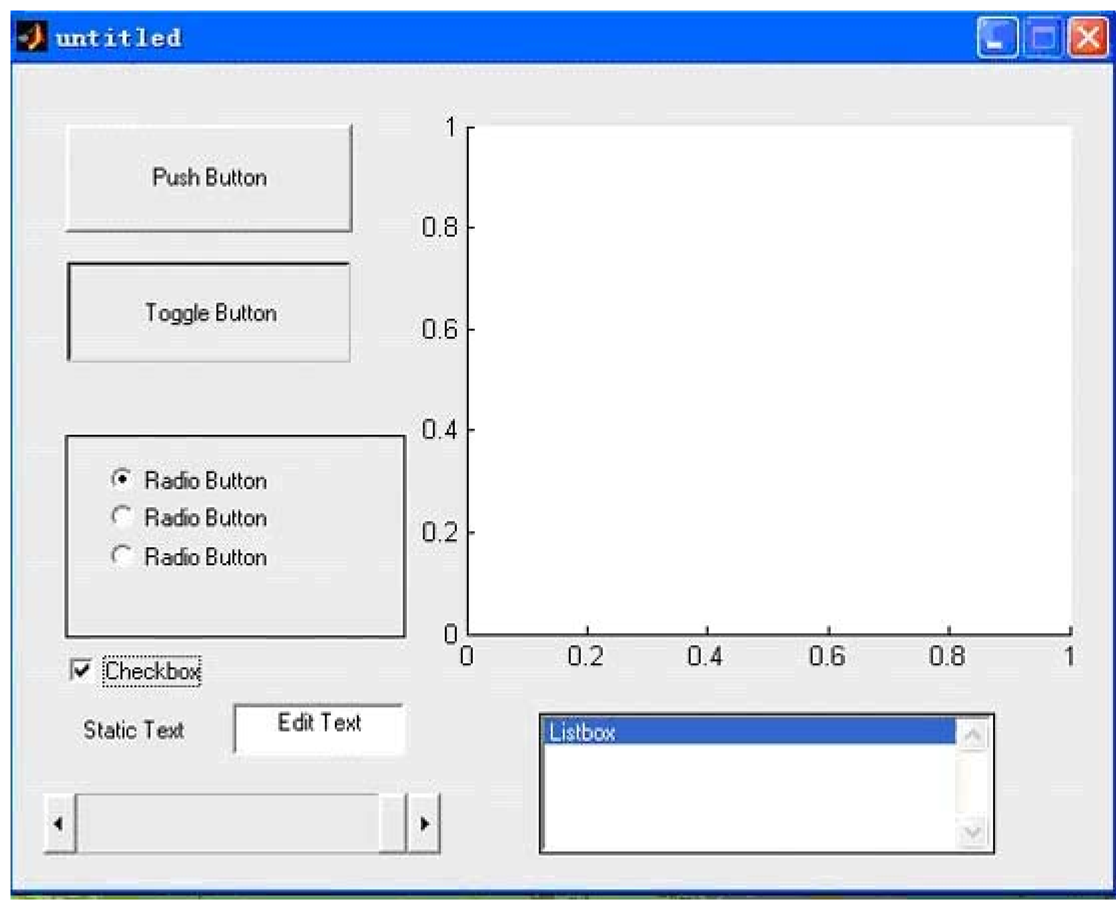


图 10.1 本图显示的是 MATLAB 用户图形界面元素的例子。按从上到下，从左向右的顺序依次为:（1）按钮（2）处于"on"状态的开关按钮。（3）在一个框架中的三个单选按钮（4）复选按钮（5）一个文本域和编辑框（6）滑动条（7）坐标系（8）列表框

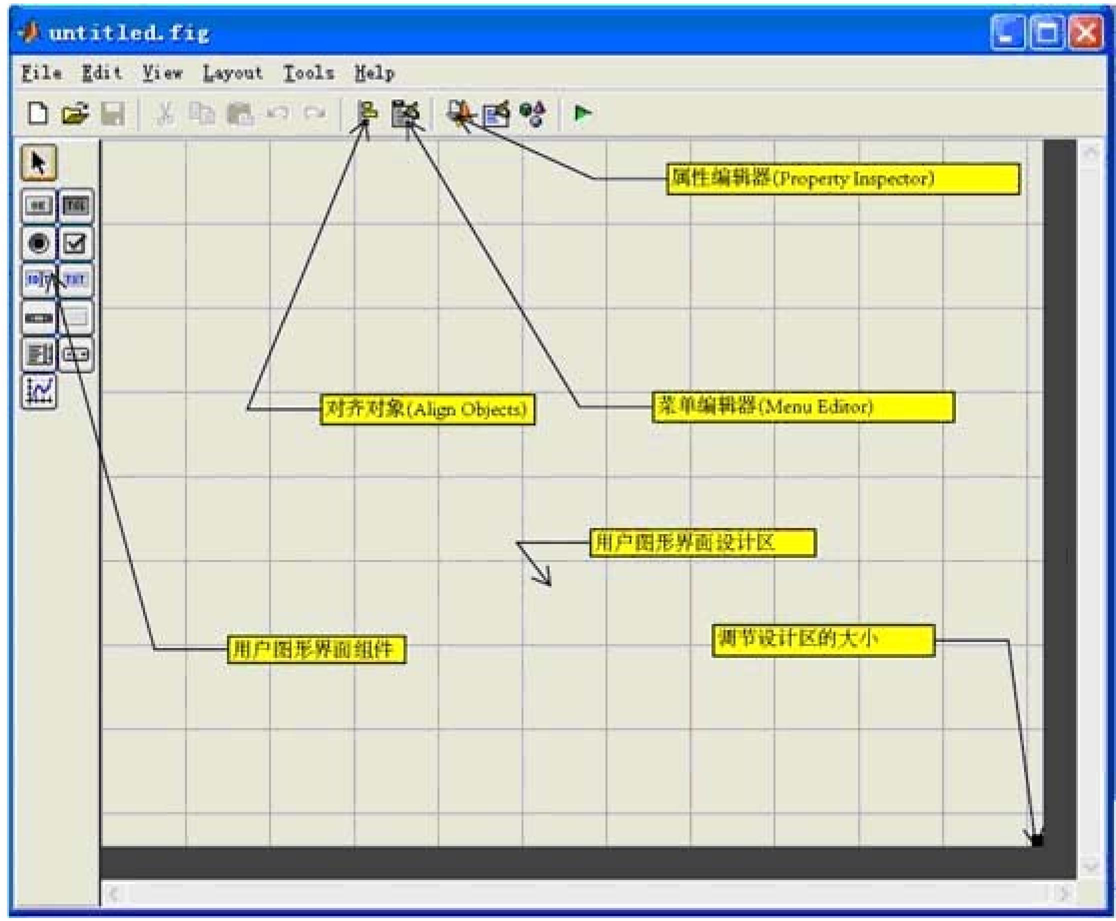
2. 调用 **MATLAB** 工具 guide 对图象中的控件进行布局。图象窗口的大小，排列和其中的控件布局都可以利用它进行控制。

3. 我们可以用 **MATLAB** 属性编辑器(property inspector)(内置于 guide)给每一个控件起一个名字(标签)，还可以设置每一个控件的其他特性，例如颜色，显示的文本等等。

4. 把图象保存到一个文件中。当文件被保存后，程序将会产生两个文件，文件名相同而扩展名相同。fig 文件包括你创建的用户图形界面，M 文件包含加载这个图象的代码和每个 GUI 元素的主要响应。

5. 编写代码，执行与每一个回叫函数相关的行为。

作为这些步骤的一个简单例子，让我们考虑一个简单的用户图形界面，它包括一个按钮和一个文本框。每单击一次按钮，文本字符串就更新一次，它用于显示用户图形界面启动后的单击总数。



第一步：这个用户图形界面是非常简单的。它包含一个简单的按钮和一个单个的文本域。这个按钮的响应为文本域中的当数字增加 1。这个用户图形界面的草图为图 10.3。

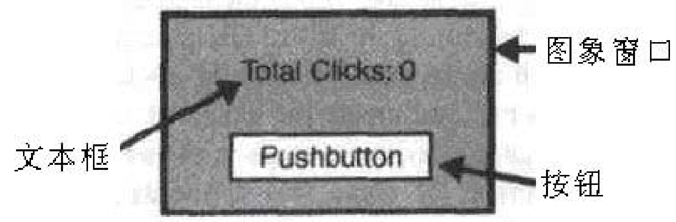


图 10.3 控件布局草图

第二步：对 GUI 控件进行布局，运行函数 guide。当运行 guide 执行时，它将产生如图 10.2 所示的窗口。

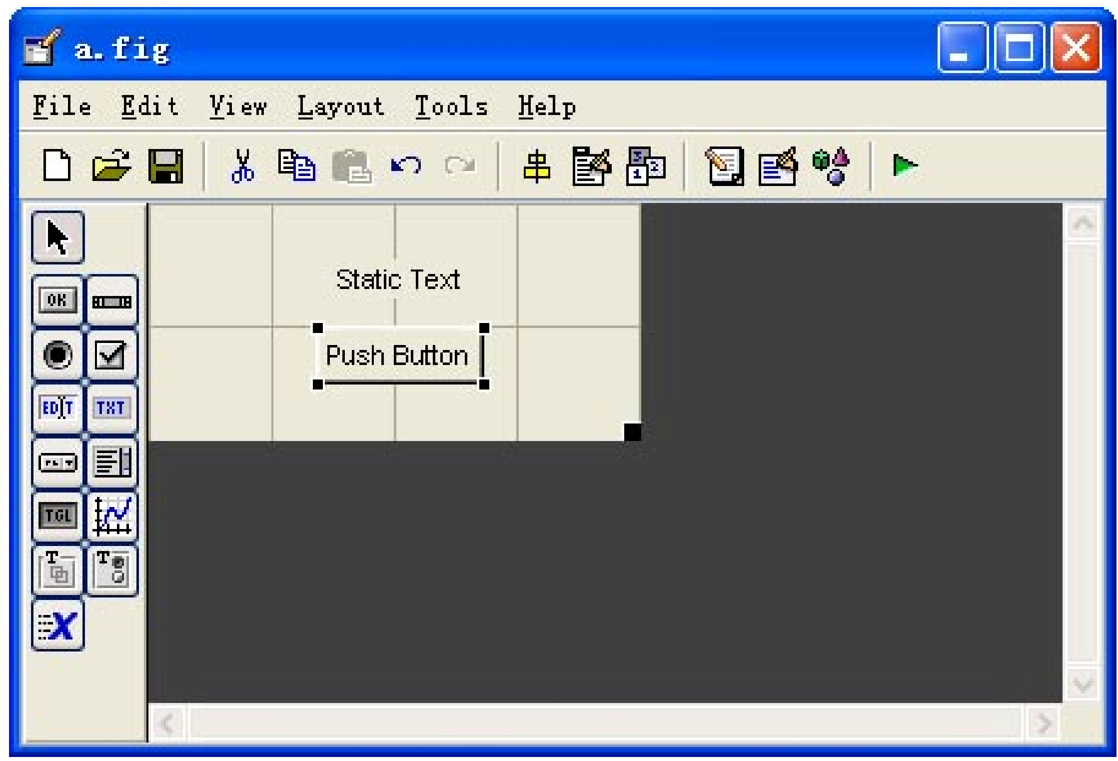


图 10.4 guide 窗口中的用户图形界面的布局

首先，我们必须设置布局的大小，它将生成最终用户图形界面的大小。我们可以通过拖动窗口右下角的小正方形调节布局区的大小和形状。然后点击“push botton”按钮然后拖动在布局区创建一个按钮。最后单击“text”按钮，然后拖动在布局区创建一个文本字符串。这些步骤产生的结果如图 10.4 所示。如果我们想让两个控件对齐的话，那么可以用对齐工具(Alignment Tool)达到此目的。

第三步：为了设置按钮的属性，右击按钮并选择“Inspect Properties”(编辑属性)。属性编辑窗口如图 10.5 所示。注意这个属性编辑器列出这个按钮的所有可以得到的属性，并允许我们改变用户图形界面的属性值。属性编辑器运行得到的结果和第九章中介绍 get 和 set 函数得到的结果相同，但是属性编辑器是一种非常容易使用的形式。

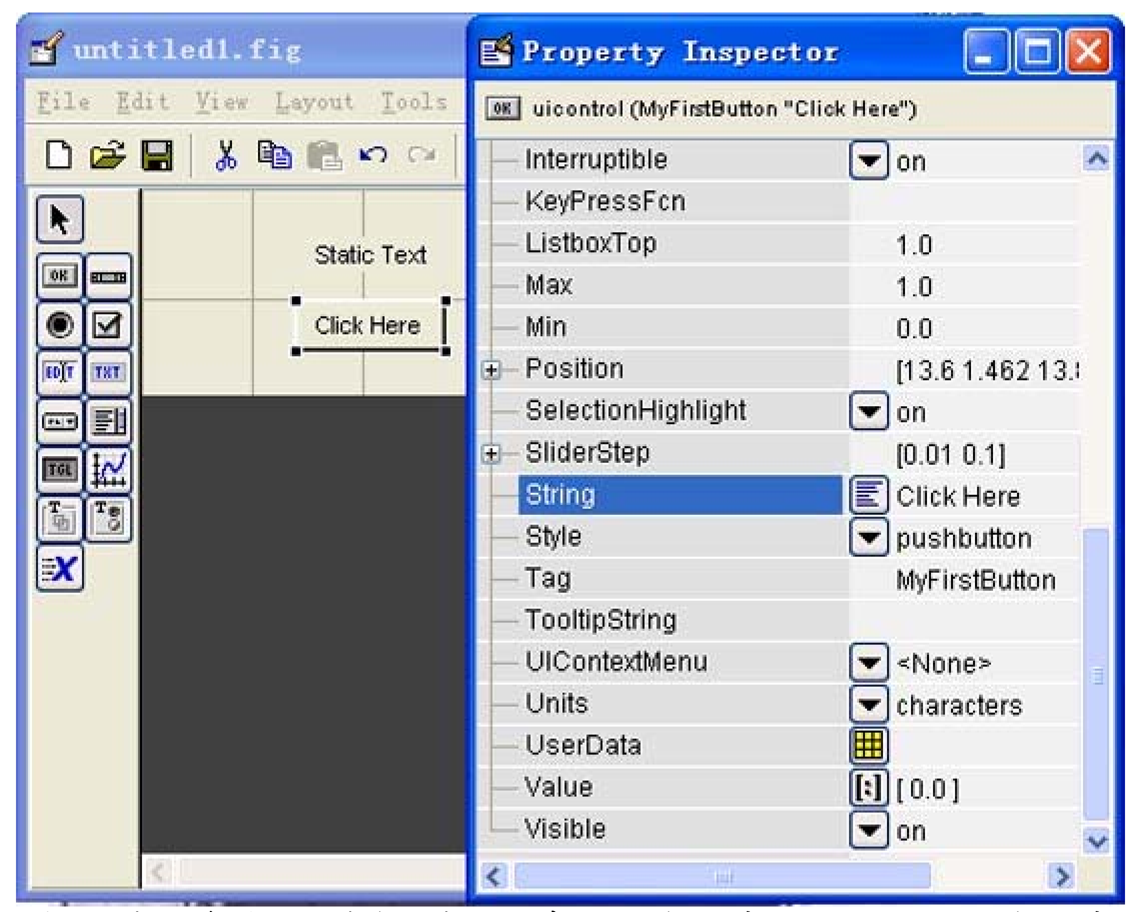


图 10.5 属性编辑器显示的按钮的属性。注意 string 被设置成"Click Here"，Tag 被设置成"MyFirstButton"

对于这个按钮来说，我们可以设置它的许多属性，例如，颜色，大小，字体，文本对齐等属性。但是有两个必须设置的属性：String 属性，它包含的属性值是所要显示的文本；Tag 属性，它的属性值为按钮的名字。在这个情况下，String 被设置为'ClickHere'，Tag 属性被设置成'MyFirstButton'。

对于文本域来说，也有两个必须设置的属性：String 属性，它包含的属性值是所要显示的文本；Tag 属性，它的属性值为文本域的名字。回叫函数需要这个名字，并更新的它的文本域。在这个情况下，String 被设置为'Total Click'，Tag 属性被设置成'MyFirstText'。经过了这些步骤，布局区如图 10.6 所示。

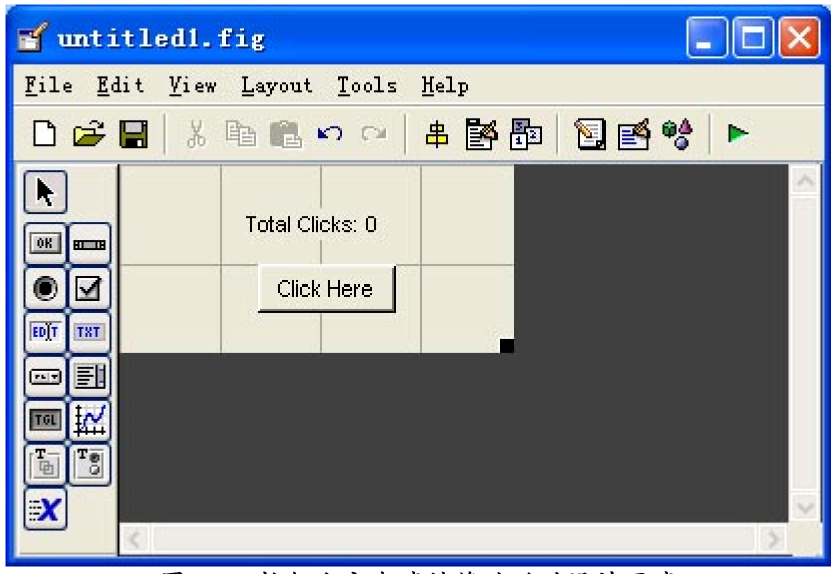


图 10.6 按钮和文本域被修改后的设计区域

我们可以通过单击鼠标在布局编辑区的空白点来调用属性编辑器，然后用属性编辑器检测并设置图象窗口的属性。即使不需要，设置图象窗口的名字是一个非常好的方法。当运行行程序时，在 name 属性中的字符串将会显示在用户图形界面的标题栏中。

第四步：我们以 MyFirstGUI 为名保存布局区。选择菜单项“File/Save as”，并键入名字 MyFirstGUI，然后单击“save”。**MATLAB** 将会产生两个文件，MyFirstGUI.flg 和 MyFirstGUI.m。图象文件由我们创建的用户图形界面构成。M 文件由加载图象文件和创建用户图形界面的代码组成，还包括每一个活动的用户图形界面组件的若干函数。

这时，我们已经拥有了一个完整的用户图形界面，但是它不能完成我们所要求的工作。在命令窗口内输入 MyFirstGUI，即可启动你的用户图形界面，如图 10.7 所示。如要你在这个主用户图形界面上单击这个按钮，在命令窗口中将会出现下面的信息。

```
MyFirstButton Callback not implemented yet.
```



图 10.7 在命令窗口中键入 MyFirstGUI,启动对应的用户图形界面

Guide 自动创建的一部分 M 文件显示在图 10.8 中。这个文件包含函数 MyFirstGUI，还有对每一个活动的用户图形界面组件执行响应的哑元子函数，如果函数 MyFirstGUI 被调用时无参数，那么这个函数将显示出包含在文件 MyFirstGUI 中的用户图形界面。如果函数 MyFirstGUI 调用时带有参数，那么函数将会假设第一个参数是子函数的名字，并用 feval 调用这个函数，把其它的参数传递给这个函数。

每一个回叫函数(call function)都控制着来自对应的用户图形界面组件的事件。如果鼠标单击事件发生在这个用户图形界面组件上，那么这个组件的回叫函数将自动被 **MATLAB** 调用。这个回叫函数的名字是这个用户图形界面组的 Tag 属性值加上字符串"_Callback"，所以，回叫函数 MyFirstButton 的名字为"MyFirstButton_Callback"。

由 guide 创建的 M 文件包括了每个活动的用户图形界面组的响应。但是这些响应只是简单的显示一条消息：回叫函数还没有被执行。

第五步：现在，我们需要执行这个按钮对应的子函数。这个函数包括一个 persistent 变量，这个变量经常被用于对点击次数进行计数。当单击次数发生在这个按钮上，**MATLAB** 将会调用带有第一个参数 MyFirstButton_Callback 的函数 MyFirstGUI。然后函数 MyFirstGUI 将会调用函数 MyFirstButton_Callback，如图 10.9 所示。

这个函数将会对单击的次数增加 1。并创建一个新的包含有这个次数的字符串，并存储在对象文本域中 String 属性的新字符串中。在这个步骤运行的函数如下所示：

```
function MyFirstButton_Callback(hObject, eventdata, handles)
% hObject      handle to MyFirstButton (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
% Declare and initialize variable to store the count
persistent count
if isempty(count)
    cout = 0;
end
% Update count
count = count + 1;
% Create new string
str = sprintf('Total Clicks: %d',count);
```

```
% Update the text field
set (handles.MyFirstText,'String',str);
```

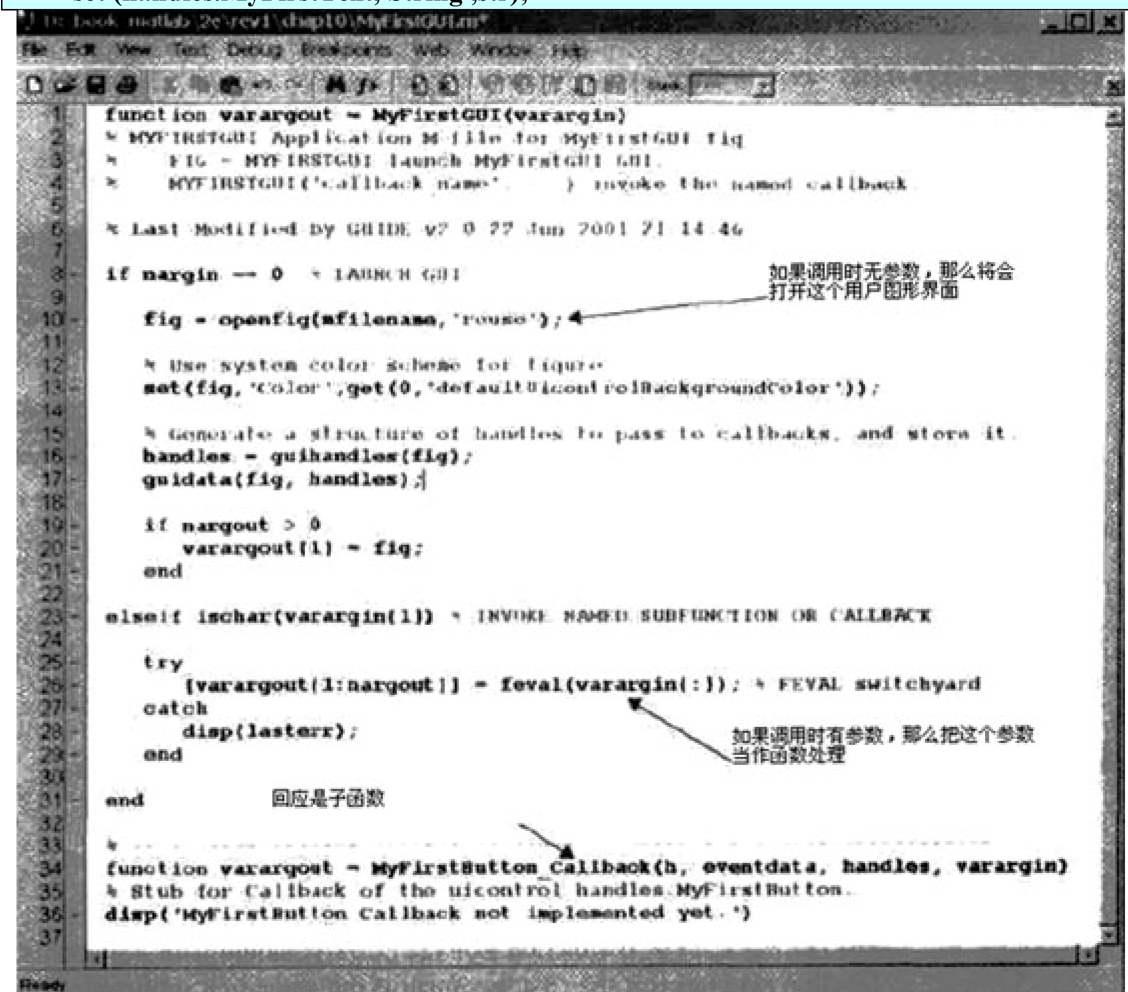


图 10.8 MyFirstGUI 的 M 文件，是由 guide 自动创建的

程序 MyFirstGUI 的事件运行过程。当一用户用鼠标在按钮进行单击，带有参数 MyFirstButton_Callback 的函数 MyFirstGUI 就会被自动调用。函数 MyFirstGUI 将会调用子函数 MyFirstButton_Callback。这个函数用来增加次数，然后把新的次数存入用户图形界面的文本域中。

注意这个函数定义了一个持久变量 cout，并把它初始化为 0。这个函数每一次调用，cout 都会增加 1，这个函数就会创建一个含有 count 的新字符串，然后，函数更新在文本域 MyFirstText 中的字符串。

在命令窗口中键入 MyFirstGUI，运行产生的程序。当用户单击这个按钮时，**MATLAB** 就会自动调用带有参数 MyFirstButton_Callback 的函数 MyFirstGUI。然后函数 MyFirstGUI 就会子函数 MyFirstButton_Callback。这个函数就把变量 count 加 1，并把这个值显示在文本域中。三次单击产生的用户图形界面如图图 10.10 所示。

好的编程习惯

用 guide 对一个新的用户图形界面进行布局，并用属性编辑器对每一个组件的初始属性进行设置，例如显示在组件上的文本，组件的颜色，还有回叫函数的名字。

好的编程习惯

用 guide 创建完一个用户图形界面后，人工编辑产生的函数，增加注释，描述这个函数的目的和组件，执行回叫函数的代码。

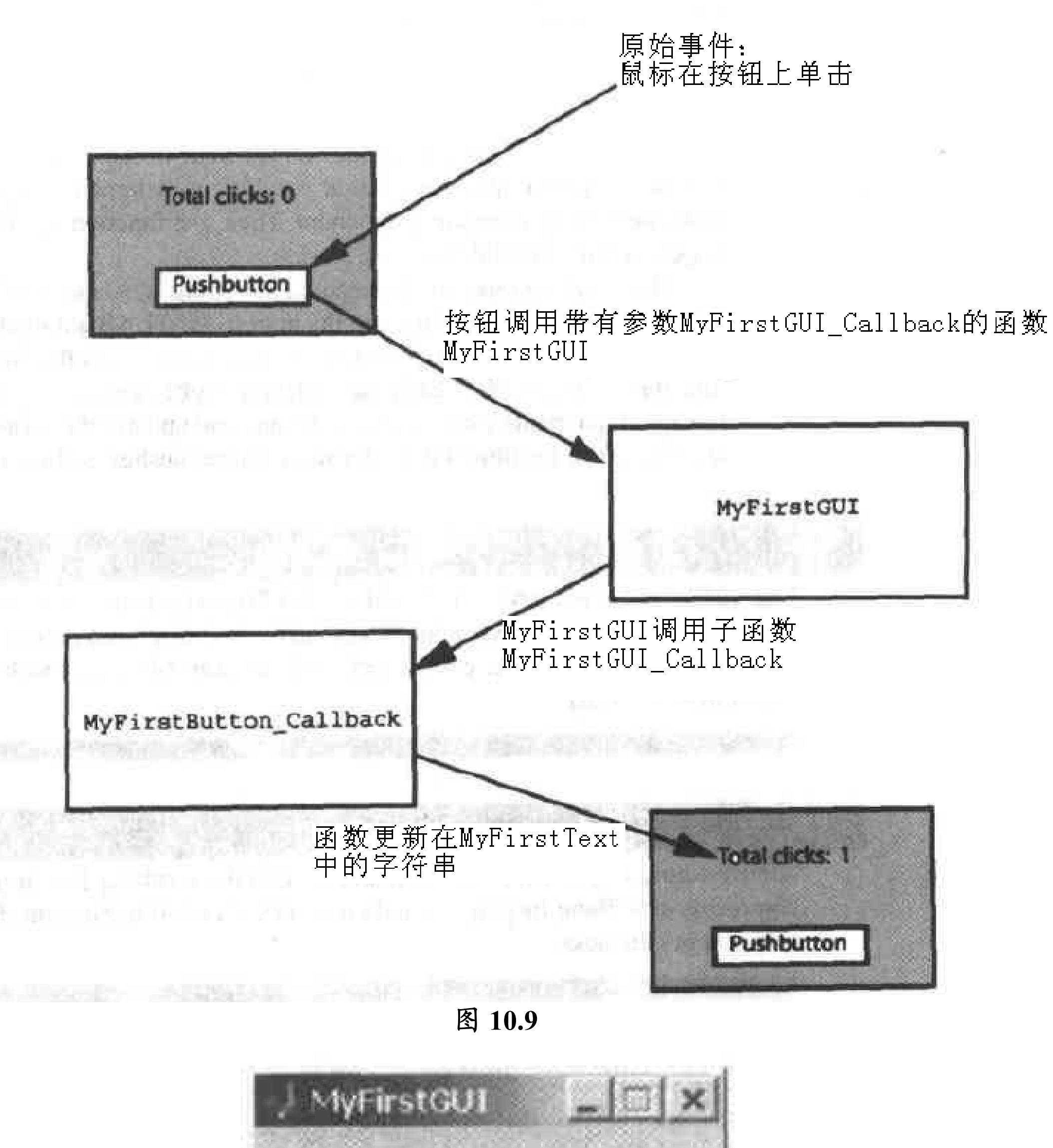


图 10.9



图 10.10 在三次单击按钮之后产生的程序

10.2.1 盖头下的一瞥

图 10.8 显示的 M 文件可以由 guide 为 MyFirstGUI 产生的。我们现在更加细致地检查这个 M 文件，以了解它是如何工作的。

首先，让我们看一下这个函数是如何声明的。注意这个函数用 vararign 来代表它的输入

参数，用 varargout 代表它的输出结果。正如我们在第七章学到的，函数 varagin 能代表任意数目的输入参数，函数 varargout 能代表可变数目的输出参数。所以，用户可以用任意数目的参数调用函数 MyFirstGUI。

● 无参数 M 文件的调用

如果用户调用无参数函数 MyFirstGUI，由 nargin 返回的值将为 0。在这种情况下，程序用函数 openfig 从图象文件 MyFirstGUI。fig 中加载用户图形界面。这个函数的形式为

```
fig = openfig(mfilename, 'reuse');
```

其中 mfilename 是指所要加载的图象文件的名字。第二个参数用于指定是一个拷贝运行指定的次数，或指定同时运行多个拷贝。如果这个参数是"reuse"，那么这个图象窗口只能有一个窗口在运行，如果调用带有"reuse"参数的函数 openfig，而且指定的图象窗口已经存了，那么原来的窗口将会被置于屏幕的顶端并被重复利用。相反地，如果这个参数为"new"，那么这个函数的多个拷贝就可以运行。如果调用带有"reuse"参数的函数 openfig，那么每一次运行都会创建一个新的拷贝。默认地，用 guide 创建的用户图形界面是参数"reuse"，所以无论在什么时侯，图象窗口只会有一个拷贝。如果你想有多个用户图形界面，你必须人工更改函数的参数。

在图象窗口被加载之后，函数将会执行下面的语句

```
set(fig, 'Color', get(0, 'defaultUicontrolBackgroundColor'));
```

这个函数用于设置图象窗口的背景色，当 **MATLAB** 运行时，背景色就是计算机默认的背景色。这个函数把这个用户图形界面的颜色设置为计算机本地窗口的颜色。所以在 windows 操作系统上编写的用户图形界面，也可以用在 UNIX 系统的计算机上，反之亦然。在不同的操作系统下，看起来非常的自然。

下面的两个语句创建了一个结构，这个结构的元素为当前窗口中所有对象的句柄，这两个语句把这个结构当作应用程序数据来存储。

```
handles = guihandles(fig);
guidata(fig, handles);
```

函数 guihandles 创建了一个结构，这个结构的元素为指定窗口中所有对象的句柄。结构的每一个元素的名字与组件的 Tag 属性相同，它的值就是每一个组件的句柄。例如在 MyFirstGUI 中返回的句柄结构为

```
>> handles = guihandles(fig)
handles =
          figure1: 99.0005
      MyFirstText: 3.0021
    MyFirstButton: 100.0007
```

在本图中有三个用户图形界面组件——图象窗口本身，一个文本域和一个按钮。函数 guidate 把句柄结构当作应用程序数据来存储，这里我们将用函数 setappdata（在第九章有介绍）。

如果有指定一个输出参数给 MyFirstGUI，那么在这里最后的一系列语将返回这个图象句柄给这个调用者。

```
if nargout > 0
    varargout{1} = fig;
end
```

● 带有参数的 M 文件的调用

如果调用带有参数的函数 MyFirstGUI，那么由 nargin 返回的值将会比 0 大。在这个种情况下，程序把第一个参数当做一个回叫函数的名字，并用 feval 执行这个函数。这个函数执行函数 naragin(1)，并把剩余的函数传递给这个函数。这种机制使子函数变为回叫函数，而这个子函数不能直接被程序其他部分直接调用。

10.2.2 一个响应子函数的结构

每一个调用子函数都有它的标准形式

function varargout = ComponentTag_Callback(h, eventdata, handles, varargin)

其中 ComponentTag 是产生响应的组件的名字(在 Tag 属性中的字符串)。子函数的参数为。

- h　　　父图象的句柄
- eventdata 当前不用的数组
- handles　由所有组件句柄构成的结构
- varargin　对回叫函数的增补函数。如果程序员需要的话，可以利用这个特性为回叫函数提供更多的附加信息。

注意每一个回叫函数都用全部的权限访问 handles 结构，所以每一个回叫函数可以修改图象文件中的每一个用户图形界面组件。我们可以可以充分利用按钮的回叫函数的结构，其中按钮的回叫函数修改了在文本域中所要显示的文本。

```
% Update the text field
set (handles.MyFirstText, 'String', str);
```

10.2.3 给图象增加应用程序数据

应用程序需要把应用程序指定的信息存储到 handles 结构中，而不存储到全局内存和持久内存中。产生的用户图形界面将更加的耐用，因为其他 **MATLAB** 程序不能突然更改全局用户图形界面数据，还因为多个相同的用户图形界面之间不相互冲突。

为了增加本地数据到 handles，在用 guide 创建了用户图形界面之后，我们必须人工地修改 M 文件。程序员，在调用应用数据到 guihandles 之后和到 guide 之前，添加应用到 handles 结构。例如，为了添加鼠标单击的次数 count 到 handles 结构中，我们将修改这个程序如下：

```
% Generate a structrue of handles to pass to callbacks
handles = guihandles(fig);
% Add count to the structure.
handles.count = 0;
% Store the structure
guidata(fig, handles);
```

当用到这个应用数据时，那么这个应用数据将会通过 handles 结构伟递给每一个回叫函数。用到 count 值的按钮回叫函数如下所示。注意如要在结构中的任意一个信息被修改了，那么我们必须把结构存储到 guidata。

```
function varargout = MyFirstButton_Callback(h, eventdata, handles, varargin)

% Update count
handles.count = handles.count + 1;

% Save the updated handles structure
guidata(h, handles);

%Create new string
str = sprintf('Total Clicks: %d', handles.count);

% Update the text field
set (handles.MyFirstText, 'String', str);
```

好的编程习惯

把 GUI 应用程序数据存储到 handles 结构中，以便任意的一个回叫函数都可以应用它。

好的编程习惯

如果你修改了 handles 结构中的任何 GUI 应用数据，确保在函数退出之前保存了调用 guidata 的结构。

10.2.4 一些有用的函数

在设计回叫函数过程中有三种特殊函数经常被使用：gcbo，gcbf 和 findobj。虽然这些函数在 **MATLAB** 6 GUIs 中的没有像以前版本那么频繁，它们还是非常有用，做为程序员，肯定会碰到。

gcbo 函数（获得回叫对象）返回产生回叫函数的对象的句柄，gcbf 函数（获得回叫图形）返回包含该对象的图形的句柄。这些函数可以被回叫函数用来确定产生回叫的对象或图形，以便可以修改图形中的对象。

findobj 函数搜索父对象中的所有子对象，查找那些指定属性具有特定值的对象，它返回任何特征匹配的对象的句柄。findobj 最常用的格式是

```
Hndl = findobj(parent, ' Property', Value');
```

其中 parent 是父对象（如图形）的句柄，Property 是要检查的属性，而 Value 是要查找的值。

例如，假设程序员要更改名称（tag）为“Button1”的按钮的文字，该程序可能先查找这个按钮，然后用下面的语句替换该文字：

```
Hndl = findobj(gcbf, 'Tag', 'Button1');
set (Hndl, 'String', 'New text');
```

10.3 对象属性

每个 GUI 对象都包含一系列可以自定义该对象的扩展属性，各种类型的对象（如图形、坐标轴，uicontrols 控件等）之间只有轻微的差别，所有类型的对象的所有属性都可以通过帮助浏览器在线找到它们的介绍文档，但是图形对象和 uicontrols 控件的一些较重要属性在表 10.2 和表 10.3 中粗略列出。

对象的属性值可以通过使用对象检查器或者是使用 get 和 set 函数进行修改，虽然对象检查器在 GUI 设计过程中可以很方便的修改属性，我们必须在程序运行过程中使用 set 和 get 函数动态地修改属性值，例如在回叫函数中进行修改。

表 10.2 图形的重要属性

属性	描述
Color	设定图形的颜色。值要么是预定义的颜色如“r”、“g”或“b”，要么是一个有 3 个元素的向量，这 3 个元素分别代表红、绿和蓝，范围从 0-1 之间。比如洋红色为[1 0 1]。
MenuBar	设定是否在图形上显示默认菜单。可以设为“figure”表示显示默认菜单，而设为“none”则删除菜单。
Name	设定要图形标题栏显示的名称
NumberTitle	设定是否在标题栏显示图形数量，可以设为“on”或者“off”。
Position	设定图形在屏幕上的位置，单位为“units”。这个值接受一个 4 元素的向量，前 2 个元素表示图形左下角的 x 和 y 坐标。而后 2 个元素则表示图形的宽和高。

表 10.2 图形的重要属性

属性	描述
SelectionType	设定鼠标在图形上的最后一次击键时选择的类型。单击返回“normal”，双击返回“open”。还有一些附加选项，请查阅 **MATLAB** 在线文档。
Tag	图形的名称，可以用来选中定位图形。
Units	用来描述图形位置的单位，可选的值为“inches”、“centimeters”、“normalized”，“points”、“pixels”或“characters”。默认为“pixels”。
Visible	设定图形是否可见，可选值为“on”或“off”。
WindowStyle	设定图形风格是普通的还是模式的（参阅对话框的讨论），可选的值为“normal”或“modal”。

表 10.3 uicontrol 控件的重要属性

属性	描述
BackgroundColor	设定对象的背景颜色。值要么是预定义的颜色如“r”、“g”或“b”，要么是一个有 3 个元素的向量，这 3 个元素分别代表红、绿和蓝，范围从 0-1 之间。比如洋红色为[1 0 1]。
Callback	设定对象被键盘或文本输入激活时被叫函数的名称和参数。
Enable	设定对象是否可选，如果不能，则对鼠标或键盘输入没有响应。可选值为“on”或“off”。
FontAngle	设定在对象上显示的字符串的角度，可选值为“normal”，“italic”和“oblique”。
FontName	设定对象上显示的字符串的字体名称。
FontSize	设定对象上显示的字符串的字号大小。默认是单位是 points。
FontWeight	设定对象上显示的字符串的字体粗细，可选值为“light”、“normal”、“demi”和“bold”。
ForegroundColor	设定对象的前景色。
HorizontalAlignment	设定对象中的字符串的水平对齐情况，可选值为“left”、“center”和“right”。
Max	设定对象 value 属性的最大值。
Min	设定对象 value 属性的最小值。
Parent	包含本对象的图形的句柄。
Position	设定图形在屏幕上的位置，单位为“units”。这个值接受一个 4 元素的向量，前 2 个元素表示对象相对于图形在左下角的 x 和 y 坐标。而后 2 个元素则表示对象的宽和高。
Tag	对象的名称，可以用来选中定位对象。
TooltipString	设定鼠标光标指到该对象时显示的帮助文本。
Units	用来描述图形位置的单位，可选的值为“inches”、“centimeters”、“normalized”，“points”、“pixels”或“characters”。默认为“pixels”。
Value	uicontrol 控件的当前值。对开关按钮、复选按钮和单选按钮来说，选中时，这个值为 max，没有选中时值为 min。对其它控件来说有不同的意义。
Visible	设定对象是否可见。可选值为“on”或“off”。

10.4 图形用户界面组件

本节概述了常见图形用户界面组件的基本特性，讨论了如何创建和使用每种组件，同时

也讨论了每种组件能产生的事件类型。本节讨论的组件有：

- 文本域
- 编辑框
- 框架
- 按钮
- 开关按钮
- 复选按钮
- 单选按钮
- 下拉菜单
- 列表框
- 滑动条

10.4.1 文本域(Text Fields)

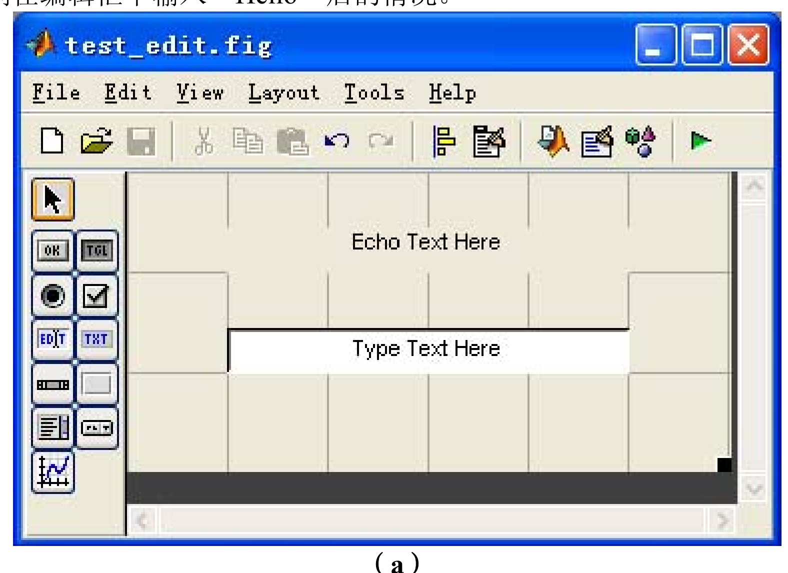
文本域是显示文本的图形对象，可以通过设定文本域的水平对齐属性改变显示文本时的对齐方式，创建时默认是水平居中。文本域是通过创建风格为“edit”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的文本域工具（TXT）把文本域添加到 GUI 中。

文本域并不产生回叫，不过文本域中显示的文本可以在回叫函数中通过设定 String 属性来更改，如 10.2 节所示。

10.4.2 编辑框(Edit Boxes)

编辑框是允许用户输入文本的图形对象，当用户在文本框中输完文本后按回车 Enter 时会产生回叫。文本域是通过创建风格为“edit”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的编辑框工具（EDIT）把编辑框添加到 GUI 中。

如图 10.11a 所示，该图是一个简单的 GUI，包含了名为“EditBox”的编辑框和名为“TextBox”的文本域各一个。当用户在编辑框中输入字符串后，它将自动调用 EditBox_Callback 函数，如图 10.11b 所示。这个函数使用 handles 结构定位编辑框获得用户的输入内容，然后再定位文本域并把字符串在文本域中显示出来。图 10.12 显示了 GUI 启动之后用户刚刚在编辑框中输入“Hello”后的情况。



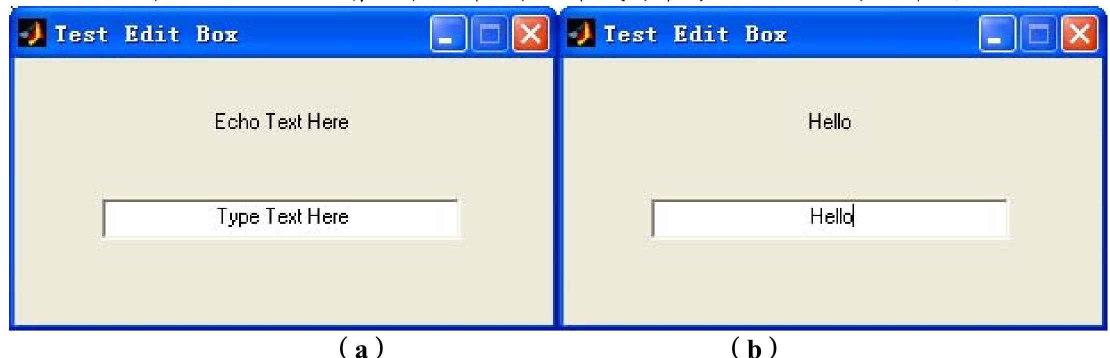
（a）

```
function EditBox_Callback(hObject, eventdata, handles)
```

```
% Find the value typed into the edit box
str = get ( handles.EditBox,'String');
% Place the value into the text field
set (handles.TextBox,'String', str);
```

（b）

图 10.11 （a）只有一个编辑框和文本域的布局（b）GUI 的回叫函数



（a）　　　　　　　　　　（b）

图 10.12 （a）由程序 test_edit 产生的 GUI （b）用户输入 Hello 并回车后的 GUI

10.4.3 框架(Frames)

框架是在 GUI 界面上显示一个矩形框，你可以把逻辑上相联系的对象用框架框起来。如图 10.1 所示，框架可以把一组单选按钮组合起来。

框架是通过创建风格为“frame”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的框架工具（）把框架添加到 GUI 中。框架不会产生回叫。

10.4.4 按钮(Pushbuttons)

按钮是当用户在其上面单击时能触发特定行为的一种组件。当用户用鼠标在其单击时，按钮会产生回叫。按钮是通过创建风格为“pushbutton”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的按钮工具（）把按钮添加到 GUI 中。

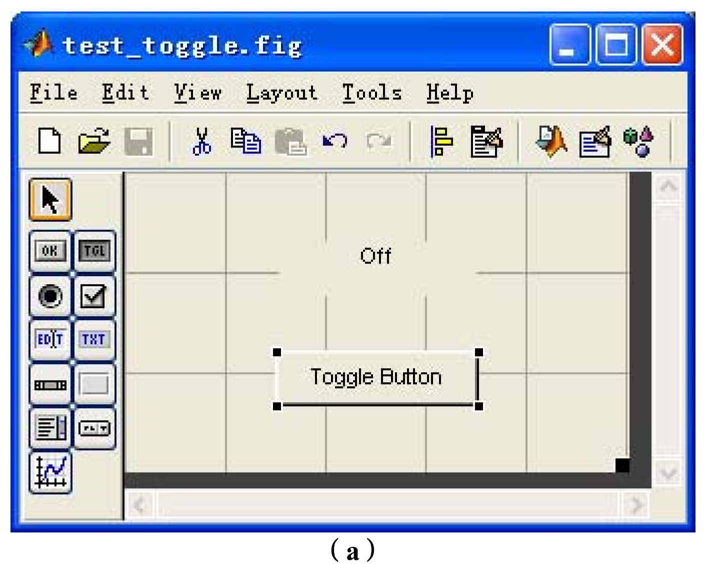
图 10.10 中的 MyFirstGUI 函数演示了如何使用按钮。

10.4.5 开关按钮(Toggle Buttons)

开关按钮是有两种状态的一类按钮：on（被按下去）和 off（没有被按下去）。当鼠标单时，开关按钮在两种状态之间切换，并且每次都产生回叫。当开关按钮为 on（被按下去）时，“Value”属性的值被设为 max（通常是 1），当为 off（没有被按下去）时，“Value”属性的值被设为 min（通常是 0）。

开关按钮是通过创建风格为“Togglebutton”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的开关按钮工具（）把开关按钮添加到 GUI 中。

图 10.13a 是一个含有开关按钮（名为“ToggleButton”）和文本域（名为“TextBox”）的简单 GUI。当用户在开关按钮上单击时，将自动调用 ToggleButton_Callback 函数（如图 10.13b 所示）。这个函数使用 handles 结构定位开关按钮并从“Value”属性获得按钮的状态，然后再定位文本域并把按钮状态在文本域中显示出来。图 10.14 显示了 GUI 启动之后用户刚刚在开关按钮上单击后的情况。

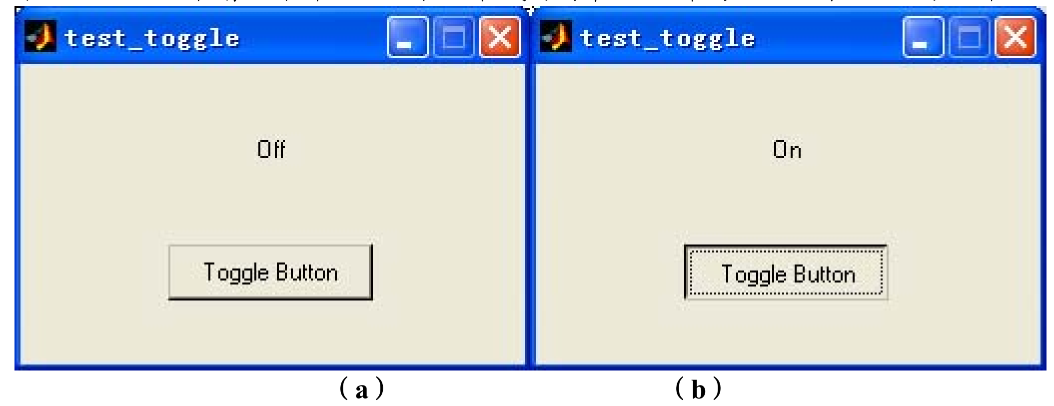


（a）

```
function ToggleButton_Callback(hObject, eventdata, handles)
% Find the state of the toggle button
state = get(handles.ToggleButton,'Value');
% Place the value into the text field
if state == 0
        set(handles.TextBox,'String','Off');
else
        set(handles.TextBox,'String','On');
end
```

（b）

图 10.13 （a）含有一个开关按钮和文本域的简单 GUI 布局 （b）本 GUI 的回叫函数



（a） （b）

图 10.14 （a）由程序 test_toggle 产生的当开关按钮为 off 时的 GUI
（b）开关按钮为 on 时的 GUI

10.4.6 复选和单选按钮(Checkboxes and Radio Buttons)

复选和单选按钮在本质上是与开关按钮一样的，除了它们的外观不同之外。与开关按钮一样，复选与单选按钮有两种状态：on 和 off。当鼠标单击时，按钮会在两种状态之间切换，每次产生回叫。当按钮状态为 on 时，复选或单选按钮的“Value”被设为 max（通常为 1），当按钮状态为 off 时，“Value”值被设为 min（通常为 0）。复选与单选按钮的形状如图 10.1 所示。

复选按钮是通过创建风格为“checkbox”的 uicontrol 控件来创建的，而单选按钮则是通过创建风格为“radiobutton”的 uicontrol 控件来创建的。可以通过使用版面编辑器中的复选

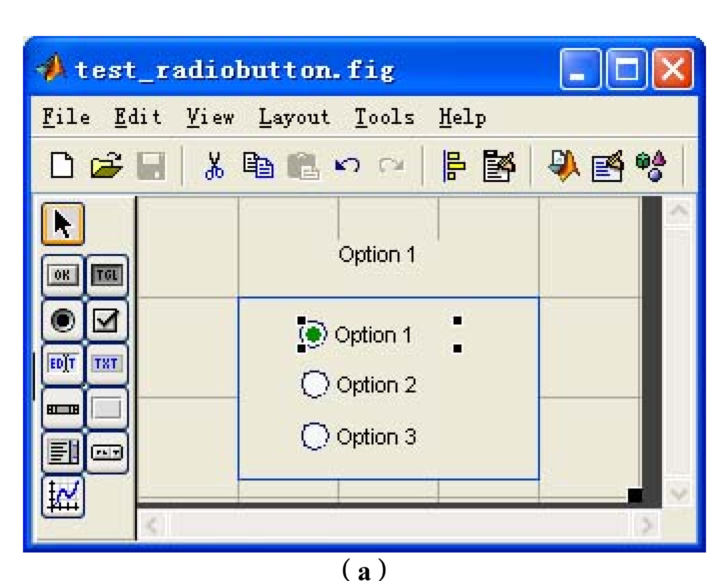
按钮工具（ ）把复选按钮添加到 GUI 中，可以通过使用版面编辑器中的单选按钮工具（ ）把单选按钮添加到 GUI 中。

复选按钮通常用来作为“开/关”选择，而单选按钮通常用在一组排它性选项中选一。

图 10.15a 演示了如何用单选按钮创建一组排它性选项。这个图形的 GUI 创建了三个单选按钮，标为“Option 1”、“Option 2”和“Option 3”，每个单选按钮拥有独自的参数，使用相同的回叫函数。

相应的回叫函数如图 10.15b 所示。当用户单击一个单选按钮时，相应的回叫函数被执行，函数在文本框中显示当前选项同，选中相应在的单选按钮，并把其它单选按钮设为未选状态。

注意 GUI 使用框架把单选按钮组合到一起，使它们看起来更像是一组。图 10.6 显示了“Option 2”被选中后的情况。



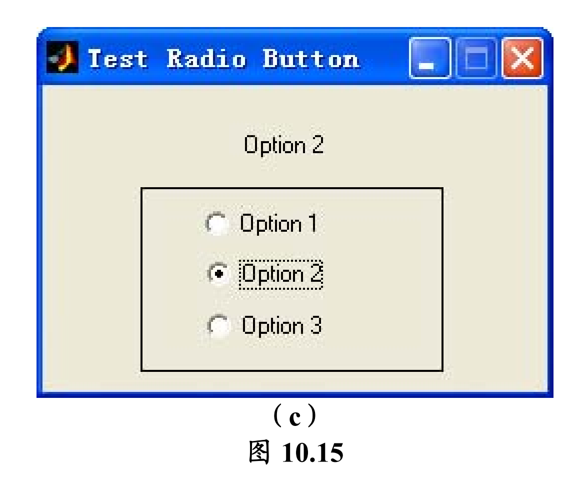
（a）

```
% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
set (handles.Label1,'String','Option 1');
set (handles.radiobutton1,'Value',1);
set (handles.radiobutton2,'Value',0);
set (handles.radiobutton3,'Value',0);

% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
set (handles.Label1,'String','Option 2');
set (handles.radiobutton1,'Value',0);
set (handles.radiobutton2,'Value',1);
set (handles.radiobutton3,'Value',0);

% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
set (handles.Label1,'String','Option 3');
set (handles.radiobutton1,'Value',0);
set (handles.radiobutton2,'Value',0);
set (handles.radiobutton3,'Value',1);
```

（b）



（c）

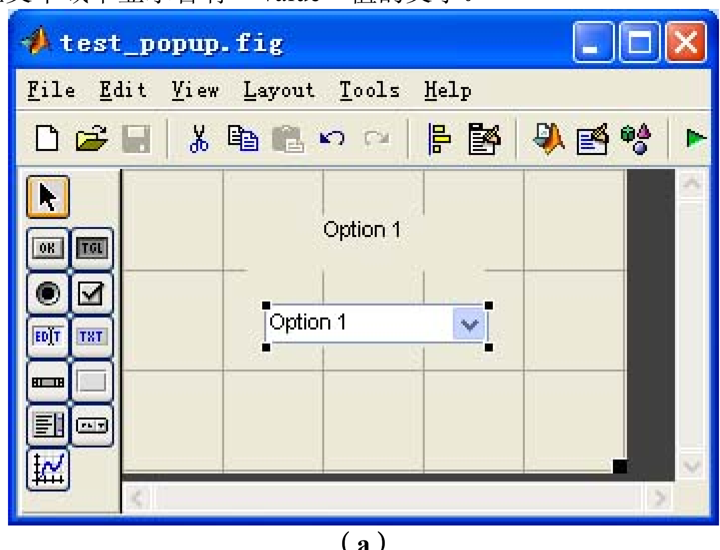
图 10.15

10.4.7 下拉菜单(Popup Menus)

下拉菜单允许用户选择一组排它性列表选项中的一项。用户可以选择的列表选项是由一个字符串胞数组指定的。“Value”属性的值指示了当前哪个字符串被选中。下拉菜单可以通过版面编辑器上的下拉菜单工具（　）添加到 GUI 中。

图 10.17a 是一个使用下拉菜单的例子。图形中的 GUI 创建了有五个选项的下拉菜单，依次标为“Option 1”、“Option 2”……

相应的回叫函数如图 10.17b 所示。回叫函数通过检查菜单“Value”参数的值取得当前选中项，然后在文本域中显示含有“Value”值的文字。



（a）

```
% --- Executes on selection change in Popup1.
function Popup1_Callback(hObject, eventdata, handles)
% Find the value of the popup menu
value = get(handles.Popup1,'Value');
% Place the value into the text field
str = ['Option ' num2str(value)];
set (handles.Label1,'String',str);
```

（b）

图 10.17 （a）含有下拉菜单和用来显示当前选择的项的文本域 GUI 布局 （b）本 GUI 的回叫函数

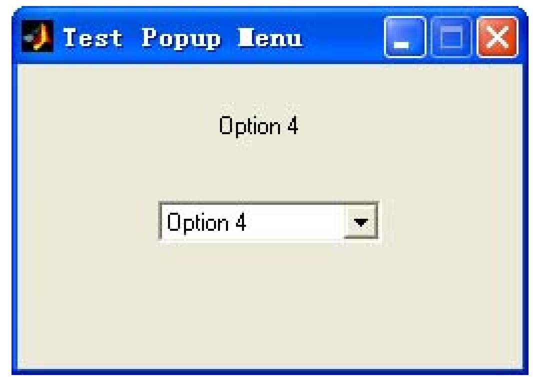


图 10.18 由程序 test_popup 产生的 GUI

10.4.8 列表框(List Boxes)

列表框可以显示多行文本并允许用户选择其中一行或多行的图形对象，如果文本的行数比刚好填满列表框还要多，则将会在列表框中出现滑动条以便用户可以上下滚动列表项。用户能够选择的文本行由一字符串胞数组指定，“Value”属性的值指示了当前哪些字符串被选中。

列表框由风格属性为“listbox”的 uicontrol 创建，用户可以使用版面编辑器中的列表框工具（）把它添加到 GUI 中。

列表框可以用来从几个可能选项中选项其中的一项。在平常的 GUI 使用中，在列表框中单击选择某项并不产生动作，而是要等到某些外部触发器（如按钮）来产生动作。不过，鼠标双击通常立即产生动作。可以通过设置图形对象的 SelectionType 属性的不同使单击与双击事件有差异：单击设置为“normal”，双击设置为“open”。

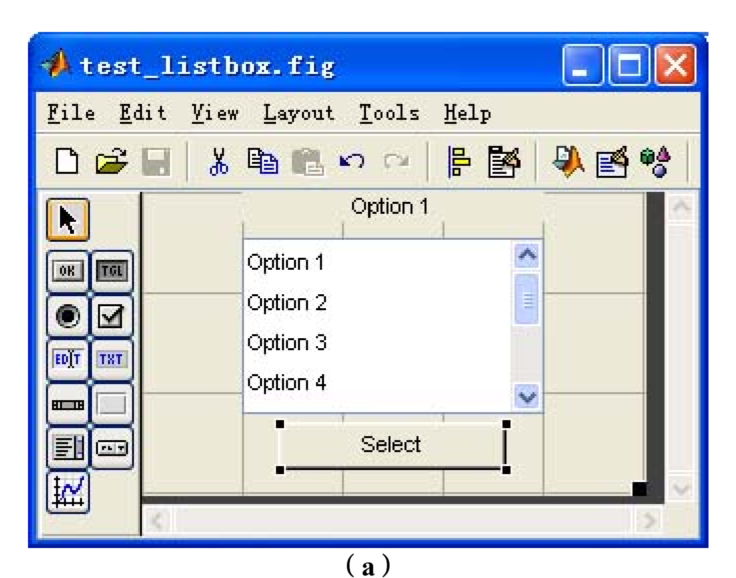
有时也允许列表框多选。如果 max 属性值与 min 属性值的差大于 1，那么就允许多选，否则则仅允许单选。

图 10.19a 是列表框仅允许单选的例子。图形中的 GUI 创建了有八个选项的列表框，这八项分别标为“Option 1”、“Option 2”……“Option 8”。此外，GUI 创建了一个按钮用来完成选择并显示选项。列表框和按钮都会产生回叫。

相应的回叫函数如图 10.19b 所示。如果列表框中的项被选中，则函数 ListBox1_Callback 将会被执行，这个函数会检查产生回叫的图形（使用 gcbf 函数），看看选择动作是单击还是双击，如果是单击，则什么也不做，双击则取得列表框中的选中值，然后在文本域中显示适当的文字。

如果是按钮被选中，则执行 Button1_Callback 函数，取得列表框中的选中值，然后在文本域中显示适当的文字。程序 test_listbox 创建的 GUI 如图 10.20 所示。

在本章练习中，你将会被要求修改本例，使得列表框可以复选。



（a）

```
% --- Executes on button press in Button1.
function Button1_Callback(hObject, eventdata, handles)
% Find the value of the popup menu
value = get (handles.Listbox1,'Value');
%Update text lable
str = ['Option ' num2str(value)];
set (handles.Label1,'String',str);

% --- Executes on selection change in Listbox1.
function Listbox1_Callback(hObject, eventdata, handles)
% if this was a double click,update the label.
selectiontype = get(gcbf,'SelectionType');
if selectiontype(1) == 'o'
        % Find the value of the popup menu
        value = get(handles.Listbox1,'Value');
        %Update text label
        str = ['Option ' num2str(value)];
        set(handles.Label1,'String',str);
end
```

（b）

图 10.19 （a）放置列表框、按钮和文本域的界面 （b）本 GUI 的回叫函数，注意单击选择和双击选择都能引发回叫。



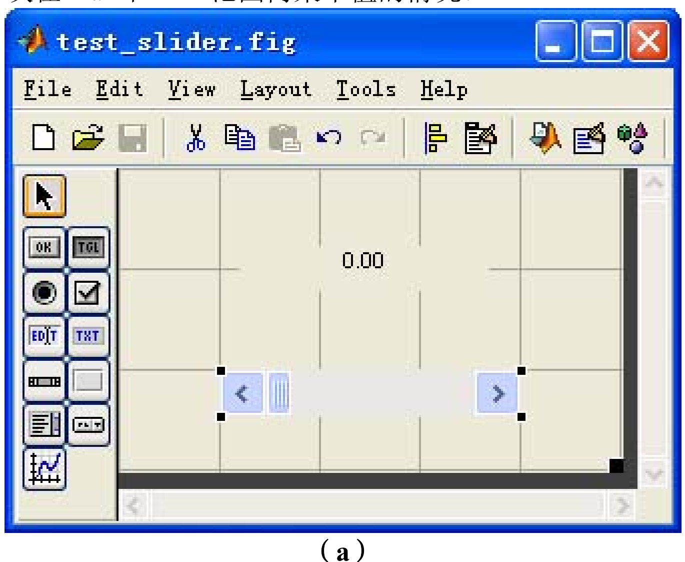
图 10.20 程序 test_listbox 产生的 GUI

10.4.9 滑动条(Sliders)

滑动条是允许用户通过用鼠标移动滑块来在最大值和最小值之间选择一个值的一种图形对象。设置到“Value”中的值在 min 和 max 之间，具体取决于滑块的位置。

滑动条是由风格为“slider”的 uicontrol 控件创建的。可以通过版面编辑器中的滑动条工具（▭）把滑动条添加到 GUI 中。

图 10.21a 是版面放置了一个滑动条和一个文本域的情况，滑动条的“min”（最小值）属性设为 0，“max”（最大值）属性设为 10。当用户拖动滑块时，将自己调用 Slider1_Callback 函数（如图 10.21b 所示），这个函数从滑动条的“Value”属性取得当前值并把它显示在文本域中。图 10.22 滑块在 max 和 min 范围内某个值的情况。



（a）

```
% --- Executes on slider movement.
function Slider1_Callback(hObject, eventdata, handles)
% Find the value of the slider
value = get(handles.Slider1,'Value');
% Place the value in the text field
str = sprintf('%.2f',value);
set (handles.Label1,'String',str);
```

（b）

图 10.21 （a）带有滑动条和文本域的简单界面 （b）本 GUI 的回叫函数

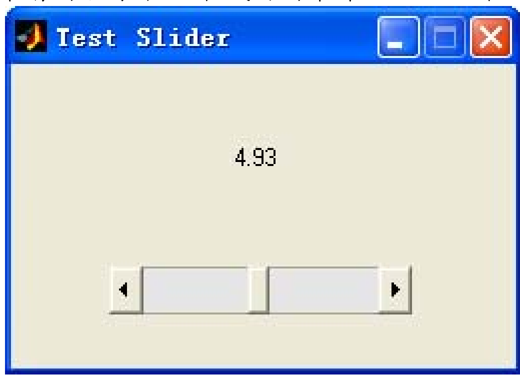


图 10.22 由程序 test_slider 产生的 GUI

例 10.1

温度转换

写一个程序，在 0-100ºC 内进行华氏温度和摄氏温度之间的相互转换，要求使用 GUI 接受数据和显示结果。程序必须有一个编辑框用来输入华氏温度，另一个编辑框用来输入摄氏温度，一个滑动条用来连续调节温度值。用户可以在编辑框中输入直接温度值，也可以通过移动滑块来输入。GUI 在中所有元素必须与值相一致。

解决方案

要创建这个程序，华氏温度需要一个文本域和编辑框，摄氏温度也同样需要一个文本域和编辑框，还需要一个滑动条。需要两个函数：一个把华氏温度转换成摄氏温度，另一个把摄氏温度转换成华氏温度。最后还需要编写回叫函数来支持用户的输入。

要转换的值范围在 32-212ºF 或 0-100ºC，因此把滑动条的范围设计 0-100 是比较合适的，并把滑动条设成摄氏温度。

这个过程的第一步是使用 guide 向导来设计 GUI，使用 guide 创建五个必需的 GUI 元素，并把它们放在合适的位置。然后使用属性编辑器实现下面的步骤：

1.为每个 GUI 元素取合适的名称并编辑相应的 Tag 值。这里取“Label1”、“Label2”、“Edit1”、“Edit2”和“Slider1”。

2.修改两处标签的“String”值为“Degrees F”和“Degrees C”。

3.把滑动条的 min 属性和 max 属性分别设为 0 和 100。

4. 保存初始值到编辑框的“String”中和滑动条的“Value”中，它们相应为“32ºF”、“0ºC”和“0”。

5.把本图形的标题栏名称（Name）设为“Temperature Conversion”。

上面的步骤完成后把 GUI 保存为 temp_conversion.fig，会产生一个图形文件和配对的 M 文件。M 文件中保存了编辑框和文本框的回叫函数的存根。界面布局如图 10.23 所示。

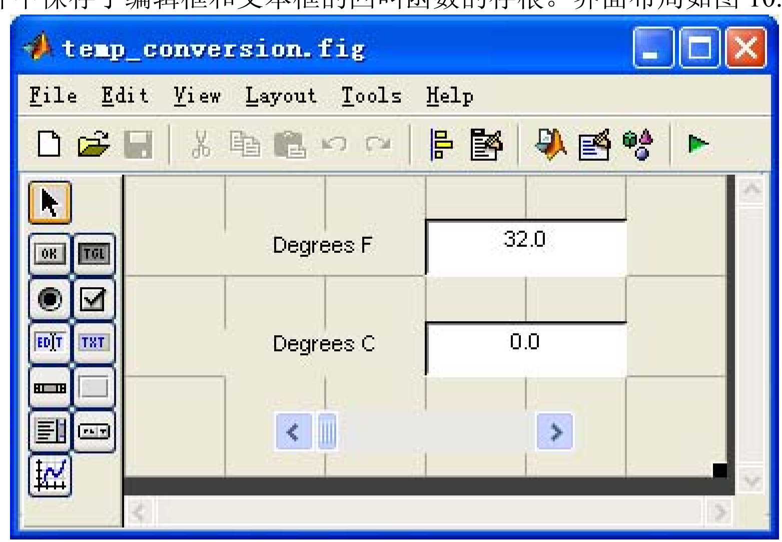


图 10.23 温度转换 GUI 界面布局

第二步是创建函数把华氏温度转换成摄氏温度。函数 to_c 把华氏温度转换成摄氏温度，它必须以下面的公式计算

$$\deg C = \frac{5}{9}(\deg F - 32) \qquad (10.1)$$

这个函数的代码是：

```
function deg_c = to_c(deg_f)
% Convert degrees Fahrenheit to degrees C.
deg_c = (5/9) * (deg_f - 32);
```

函数 to_f 把摄氏温度转成成华氏温度，用下在的公式计算

$$\deg F = \frac{9}{5}(\deg C) + 32 \tag{10.2}$$

这个函数的代码如下：

```
function deg_f = to_f(deg_c)
% Convert degrees Celsius to degrees Fahrenheit
deg_f = (9/5) * deg_c + 32;
```

最后，我们必须编写回叫函数把它绑在一起。这些函数必须能对三个按钮作出反应。（注意我们既要难对用户在编辑框的输入做出更新，也要以一致的格式来显示数据，同时，如果用户输入的值落在允许范围之外，还必须能够更正它。）

这里还有另外一个因素要考虑：我们在编辑框中输入的值是字符串 strings，但我们要以数字来对待它。如果用户在编辑框输入 100，则实际上得到的是字符串“100”，并不是数字 100。回叫函数必须把字符串转换成数字，以便用来计算。这种转换用 str2num 函数来完成，它把字符串转换成数字。

同时，回叫函数必须限制用户输入的值在有效范围之内，即 0-100ºC 和 32-212ºF 之间。回叫函数如图 10.24 所示。

```
function Edit1_Callback(hObject, eventdata, handles)
% Update all temperature value
deg_f = str2num(get(hObject,'String'));
deg_f = max([32 deg_f]);
deg_f = min([212 deg_f]);
deg_c = to_c(deg_f);

% Now update the fields
set (handles.Edit1,'String',sprintf('%.1f',deg_f));
set (handles.Edit2,'String',sprintf('%.1f',deg_c));
set (handles.Slider1,'Value',deg_c);

function Edit2_Callback(hObject, eventdata, handles)
% Update all temperature value
deg_c = str2num(get(hObject,'String'));
deg_c = max([0 deg_c]);
deg_c = min([100 deg_c]);
deg_f = to_f(deg_c);

% Now update the fields
set (handles.Edit1,'String',sprintf('%.1f',deg_f));
set (handles.Edit2,'String',sprintf('%.1f',deg_c));
set (handles.Slider1,'Value',deg_c);

% --- Executes on slider movement.
function Slider1_Callback(hObject, eventdata, handles)
% Update all temperature values
deg_c = get(hObject,'Value');
deg_f = to_f(deg_c);

% Now update the fields
set (handles.Edit1,'String',sprintf('%.1f',deg_f));
set (handles.Edit2,'String',sprintf('%.1f',deg_c));
set (handles.Slider1,'Value',deg_c);
```

图 10.24 温度转换 GUI 的回叫函数

程序完成，执行并分别向文本框或滑动条输入几个不同的值试试，注意使用范围之外的值试试。它是否正确执行了？

10.5 对话框

对话框是一种通常用来显示信息或从用户处获得输入的特殊图形，对话框通常用来显示出错、提供警告、提问问题或获取输入。它们也用来选取文件或打印机属性。

对话框可以是有模式的或无模式的。模式对话框在它消失之间不允许程序中的其它窗口被访问，而无模式对对话框则不存在这些限制。模式对话框典型的应用是用来显示错误或警告，而这些信息通常是需要引起重视，不可忽略的。默认地，大多数对话框是无模式的。

MATLAB 包含了很多类型的对话框，大多数对话框已经在表 10.4 中列出。这里我们只讨论其中的几种，关于其它对对话框的内容请参阅 **MATLAB** 在线文档。

表 10.4 挑选的对话框

属性	描述
dialog	创建一个通用对话框
errordlg	在对话框中显示错误信息，用户必须点击 OK（确定）按钮才能继续。
helpdlg	在对话框中显示帮助信息，用户必须点击 OK（确定）按钮才能继续。
inputdlg	显示需要输入数据，并接受输入的数据。
printdlg	显示打印机选择对话框。
questdlg	提问。这种对话框可以包含二到三个按钮，默认是 Yes（是），No（否）和 Cancel（取消）。
uigetfile	显示打开文件对话框。这类对话框允许用户选择要打开的文件，但并不打开文件。
uiputfile	显示保存文件对话框。这类对话框允许用户选择要保存的文件，但并不保存文件。
uisetcolor	显示颜色选择对话框。
uisetfont	显示字体选择对话框。
warndlg	在对话框中显示警告信息，用户必须点击 OK（确定）按钮才能继续。

10.5.1 错误和警告对话框

错误与警告对话框有相似的调用参数与行为，实际上，这两种对话框仅仅是显示的图标不同而已。最常见的调用格式如下：

```
errordlg(error_string,box_title,create_mode);
warndlg(warning_string,box_title,create_mode);
```

其中 error_string 或 warning_string 就是要显示给用户的信息，box_title 是对话框的标题，create_mode 可以是“modal”或“non_modal”，这取决于你要何种对话框。

例如，下面的语句创建一个用户无法忽略的模式对话框，这个语句产生的对话框如图 10.25 所示。

```
errordlg('Invalid ipnut values!','Error Dialog Box','modal');
```

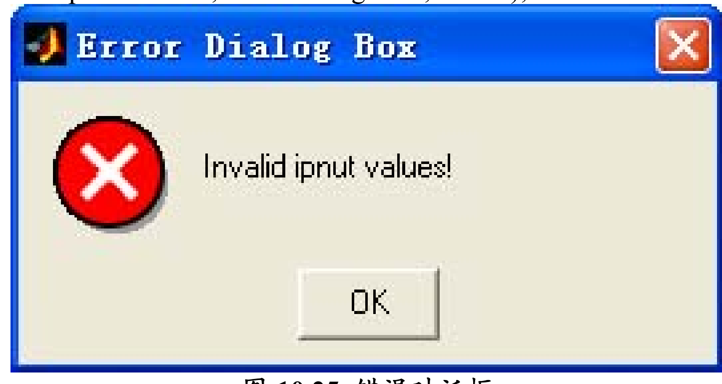


图 10.25 错误对话框

10.5.2 输入对话框

输入对话框提示用户输入程序需要的一个或多个值，可以用下面格式创建：

```
answer = inputdlg(prompt)
answer = inputdlg(prompt, title)
answer = inputdlg(prompt, title, line_no)
answer = inputdlg(prompt, title, line_no, default_answer)
```

这里的 prompt 是一个字符串数组，数组的每个元素就是要求用户回答的问题，title 指定对话框的标题，line_no 限定用户输入的行数，最后，default_answer 是一个胞数组——包含了具体某个问题用户没有回答时的默认答案。注意有多少个问题，就必须提供多少个默认答案。

当用户单击对话框上的 OK 按钮时，答案将在字符串胞数组的形式保存到变量 answer 中。

作为输入对话框的例子，假设我们要用输入对话框允许用户指定图形的位置，下面的代码将能做到：

```
prompt{1} = 'Starting x position:';
prompt{2} = 'Starting x position:';
prompt{3} = 'Starting x position:';
prompt{4} = 'Starting x position:';
title = 'Set Figure Position';
default_ans = {'50', '50', '180', '100'};
default = inputdlg(prompt,title,1,default_ans);
```

结果如图 10.26 所示。

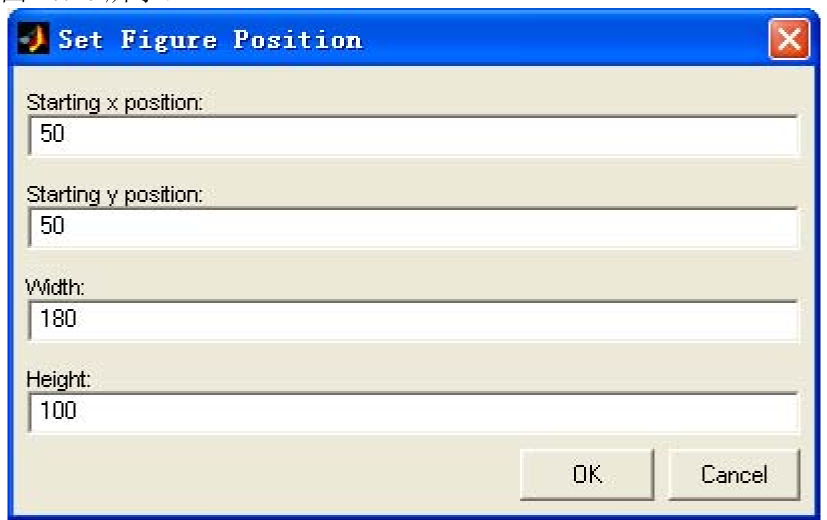


图 10.26 输入对话框

10.5.3 打开与保存对话框

uigetfile 和 uisetfile 对话框是设计用来允许用户交互地选择要打开或保存的文件，这些对话框返回文件名及路径，但并不实际读取或保存它。程序员负责写额外的代码。

这两个对话框的形式如下：

```
[filename, pathname] = uigetfile(filter_spec, title);
[filename, pathname] = uisetfile(filter_spec, title);
```

参数 filter_spec 是指定在对话框中显示的文件类型的字符串，如“*.m”、“*.mat”等等。参数 title 指定对话框的标题。对话框执行后，filename 包含了所选择的文件名，而 pathname

包含了文件的路径。如果取消对话框，则 filename 被设为 0。

下面的脚本文件演示了如何使用这些对话框，它提示用户输入 MAT 文件名，读取文件的内容，图 10.27 是在 Windows Xp 系统中运行的结果（不同的操作系统显示的界面会有不同）。

```
[filename, pathname] = uigetfile('*.mat', 'Load MAT File');
if filename ~= 0
    load([pathname filename]);
end
```

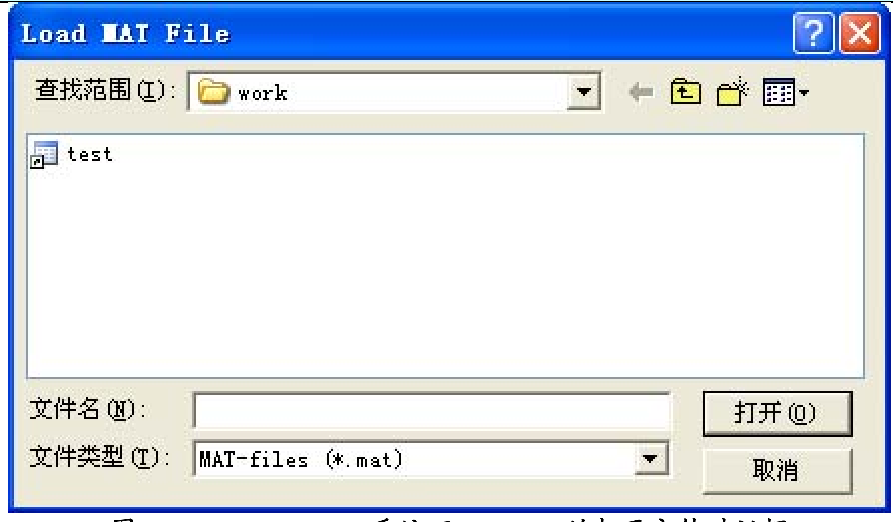


图 10.27 Windows xp 系统下 uigetfile 的打开文件对话框

好的编程习惯

在基于 GUI 编程中，使用对话框来提供信息或要求输入数据，如果信息紧迫且不可忽略，则把对话框设为模式对话框。

10.6 菜单

菜单也可以添加到 **MATLAB** 的 GUI 中。使用菜单可使 GUI 显示界面上没有附加组件用户即可选择行动。

MATLAB 中有两种菜单：在图形界面顶部的菜单栏下拉的**标准菜单**与在某对象上右击的弹出的**上下文菜单**。本节我们将学习如何创建这两类菜单。

标准菜单是由 uimenu 对象创建的，菜单中的每一项及子菜单都是一个独立的 uimenu 对象，这些 uimenu 对象与 uicontrol 对象相似，有许多相同的属性，例如 Parent，Callback，Enable 等等，一些 uimenu 较重要的属性在表 10.5 中列出。

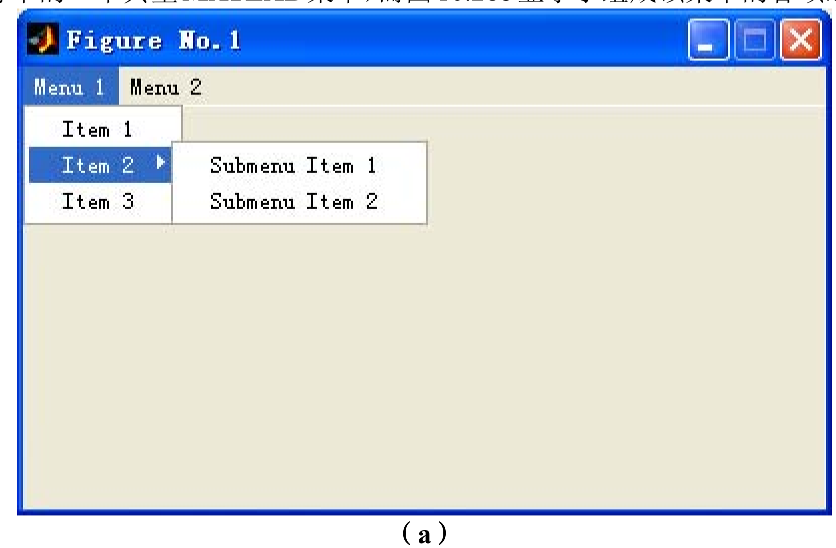
表 10.5 重要的 uimenu 属性

属性	描述
Accelerator	指定键盘上某个字符与该菜单项相当，用户可以使用键盘组合键 CTRL + key 激活该项。
Callback	设定该菜单项激活时调用的函数的名称与参数。如果该项是一个子菜单，则该函数在子菜单显示之前执行；如果该项没有子菜单，则该函数在鼠标释放时执行。
Checked	如果该项为 on，则在该项左边会有一个选择标记，这种属性可以用来指示菜单项在两种状态之间转换在情况，可选值为“on”和“off”。
Enable	设定该菜单项是否可选，如果不能，则不会对任意鼠标单击与

表 10.5 重要的 uimenu 属性

属性	描述
	键盘加速键做出响应，可选值为“on”或“off”。
Label	设定在菜单项上显示的文本，可以用“&”符号指定本项的键盘助记键盘并在菜单上显示出来。例如，字符串“&File”将在菜单上显示“File”并对 F 键作出响应。
Parent	本菜单项的父对象的句柄。父对象可以是一个图形或另一个菜单项。
Position	设定菜单项在菜单栏或菜单上的位置，1 代表在顶级菜单中的最右边或在子菜单中的最上边。
Separator	如果属性为“on”，则在这项上画一条分隔线，可选的值为“on”或“off”。
Tag	菜单项的名称，可用来定位菜单项。
Visible	设定本菜单项是否可见，可选值为“on”或“off”。

每个菜单项都是都隶属于某个父对象，而图形本身是一个最顶端的菜单。所有的隶属于同一父对象的 uimenus 显示在相同的菜单上，各项叠起来形成一个树形子菜单。图 10.28a 显示了运行中的一个典型 **MATLAB** 菜单，而图 10.28b 显示了组成该菜单的各项之间的关系。



（a）

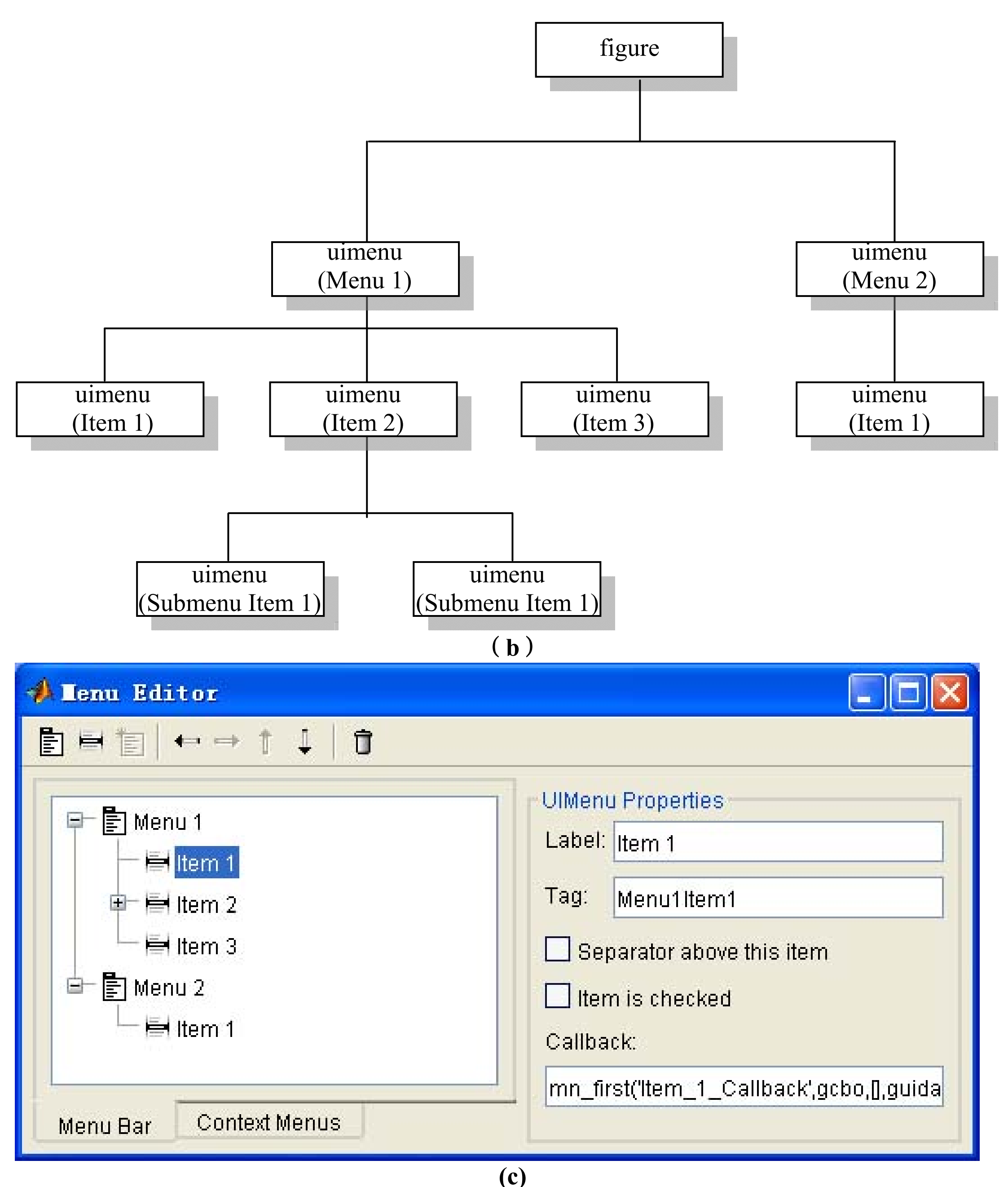


图 10.28 （a）典型的菜单结构 （b）创建菜单的 uimenu 项之间的关系 （c）菜单编辑器中的结构

MATLAB 使用菜单编辑器创建菜单，可以通过上界面编辑器向导顶端的图标打开。图 10.28c 显示了产生菜单上各项的结构。附加属性在表 10.5 中列出。菜单编辑器可以用属性编辑器设置（propedit）。

顶级上下文菜单由 uicontextmenu 对象创建，上下文菜单中的下一级菜单项则由 uimenu 对象创建，上下文菜单基本上与标准菜单相同，除了可以隶属于任何对象（坐标轴，线条，文本，图形等等）外。表 10.6 列出了 uicontextmenu 一些重要的属性。

10.6.1 禁用默认菜单

每个 **MATLAB** 图形默认都会出现一组标准菜单，如果你要删除这些菜单并创建自己的菜单，首先就必须把默认菜单关闭掉，是否显示默认菜单是由图形的 Menubar 的属性控制

的。或选值为“figure”和“none”。如果设为“figuire”，那么将显示菜单，如果设为“none”，默认菜单将会抑制。创建 GUIs 时你可以使用属性编辑器来设置。

表 10.6 uicontextmenu 几个重要属性

属性	描述
Callback	设定上下文菜单激活时调用的函数的名称和参数。在菜单显示之前执行。
Parent	上下文菜单的父对象的句柄。
Tag	上下文菜单的名称，可以用来定位它。
Visible	设定上下文菜单是否可见。这个属性会被自动设置，通常不需要修改。

10.6.2 创建自定义菜单

为 GUI 创建自定义标准菜单基本上分三步处理：

1.第一，在菜单编辑器中创建一个新的菜单结构。使用菜单编辑器定义结构，给每个菜单项填上 Label 标签和独一的 Tag 名称，同时也可以为菜单项手动创建回叫字符串，这需要些技巧——为菜单项创建回叫的最好办法是对 uicontrol 自动创建的回叫进行研究，并把它当成例子学习。uimenu 回叫字符串的正确形式如下：

MyGui('MenuItemTag_Callback', gcbo, [], guidata(gcbo));

这里你要把 MyGui 替换成你自己的 GUI 名称，把 MenuItemTag 设为菜单项的名称。

2.第二，使用属性编辑器（propedit）编辑每个菜单项的属性，最重要的几个需要设置的是 Callback，Label 和 Tag，这些都可以通过菜单编辑器进行设置，因此通常并不需要属性编辑器。当然，如果你要设置表 10.5 中列出的其它属性，就必须使用属性编辑器了。

3.第三，为菜单项编写代码实际回叫函数所必须完成的功能。你必须为菜单项手动创建回叫函数。

在本章节末尾，将以一个例子演示创建菜单的过程。

编程隐患

与 GUI 对象不同，MATLAB 并不自动为菜单项创建回叫字符串与存根，必须手动完成。

编程隐患

菜单项的属性中只有 Label，Tag，Callback，Checked 和 Separator 等属性可以在菜单编辑器中设置，如果要设置其它属性，就必须使用图形中的属性编辑器，选择正确的菜单项进行编辑。

10.6.3 加速键与键盘助记键

MATLAB 菜单支持键盘加速键与助记键。加速键是“CTRL+key”组合键，不需要先打开菜单就能使某菜单项执行。例如，加速键“o”可能分配给菜单中的“File/Open（文件/打开）”，如果这样，键盘组合键“CTRL+o”将会使“File/Open”项的回叫函数执行。

有一些组合键盘为操作系统所保留，因操作系统（PC 或 UNIX）而异，查阅 **MATLAB** 在线文档，看看哪些组合键适合你的计算机。

加速键在 uimenu 对象的 Accelerator 属性中设定。

键盘助记键仅是一个字母，在菜单被打开后敲击该字母就会执行该菜单项。分配菜单项的助记键被加以下划线。如果是顶级菜单，则必须按住键盘的 ALT 键再同时按下该字母，一旦顶级菜单已经打开，则只需敲击该字母就能执行该项。

图 10.29 演示了键盘助记键的使用。File（文件）菜单使用 ALT＋f 打开，一旦菜单打开，

只需敲“x”即可执行 Exit（退出）项。

助记键是通过在 Label 属性中的目标字母前面加上“&”字符来定义的。“&”不会被显示出来，后面是字符会被加以下划线，它就成了助记键。例如，图 10.29 中的 Exit（退出）菜单项的 Label 属性值是“E&xit”。

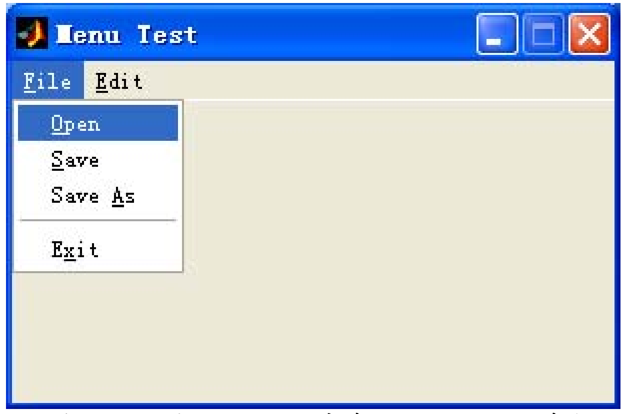


图 10.29 通过 ALT+f 打开显示的菜单，要退出只需单击“x”即可

10.6.4 创建上下文菜单

上下文菜单与创建普通菜单的方式相同，除了顶级菜单项是一个 uicontextmenu 外。uicontextmenu 的父对象必须是一个图形。但上下文菜单也可以与任何图象对象绑定起来并对鼠标右键做出响应。上下文菜单通过在菜单编辑器选择“Context Menu”项来创建。一旦上下文菜单创建后，就可以在它下面创建任意数量的菜单项。

要把上下文菜单与特定对象绑定起来，你必须把对象的 UiContextMenu 属性设置成 uicontextmenu 的句柄。这通常通过属性编辑器来完成，但也可以使用下面的 set 命令来做。假如 Hcm 是一个上下文菜单的句柄，下面的语句将把这个菜单与用 plot 命令创建的线条绑定起来。

```
H1 = plot(x,y);
set (H1, 'UIContextMenu',Hcm);
```

在下面的例子中，我们创建一个上下文菜单，并把它与一个图对象联合起来。

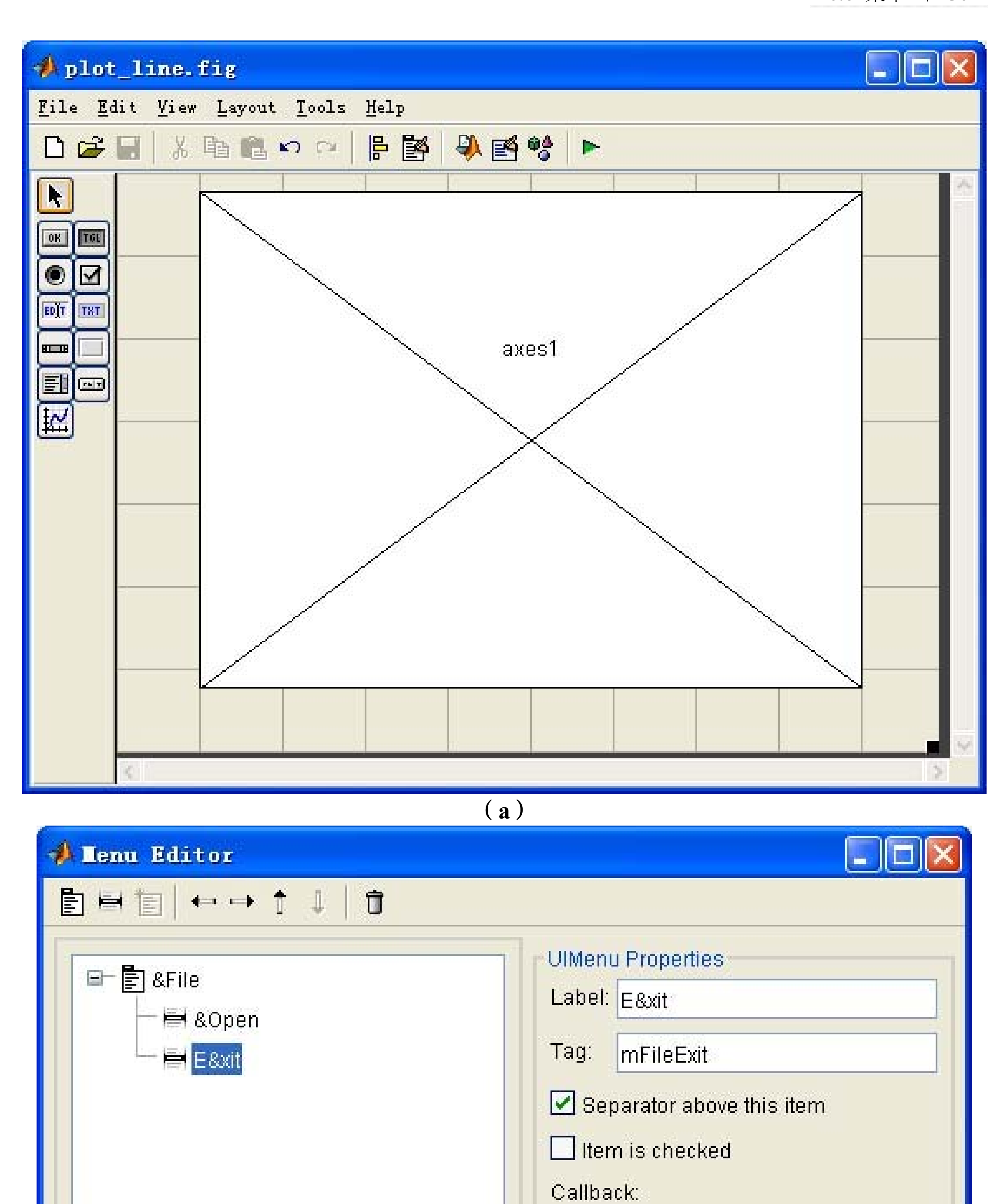
例 10.2 绘制数据点

编写程序，打开用户指定的数据文件，然后把文件中数据绘点画线。程序必须包含一个文件（File）菜单，有打开（Open）和退出（Exit）项，还必须包含一个绑定到线条上的上下文菜单，菜单的内容可以改变线条的风格。假设文件的数据是以（x，y）对的形式出现，每行一对。

解决方案

本程序必须包含一个有打开（Open）和退出（Exit）菜单项的标准菜单和一个用来绘制数据的坐标轴，也必须包含一个用来设定不同线条风格的上下文菜单，还必须把上下文菜单绑定到描绘的线条中，上下文菜单项中应该包含实线、虚线、点线和虚点线等风格。

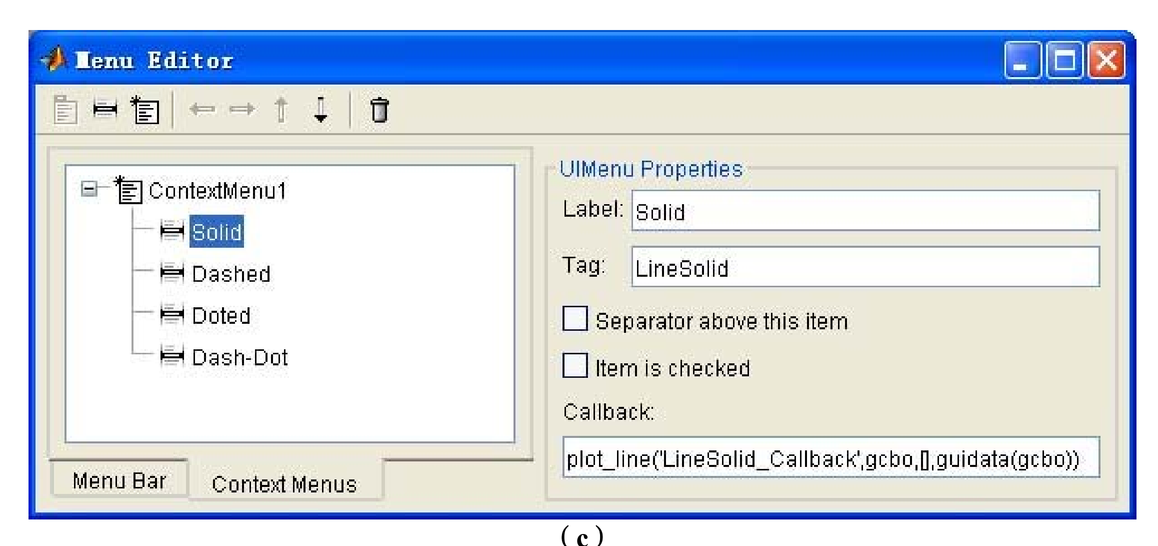
创建本程序的第一步是使用向导创建所需要的 GUI，本例中界面只有一组坐标轴（见图 10.30a）。然后，我们用菜单编辑器创建文件菜单，这个菜单包含了 Open（打开）和 Exit（退出）项，如图 10.30b。注意我们必须使用菜单编辑器设置 Label，Tag 和每个菜单项的 callback 字符串。还必须为 File（文件）指定助记键“F”，为 Open（打开）指定“O”及为 Exit（退出）指定“x”，并在 Exit 与 Open 之间设置分隔符，如图 10.30 所示。



（a）

（b）

图 10.30 （a）plot_line 的界面 （b）菜单编辑器中的文件菜单



（c）

图 10.30 续（c）菜单编辑器中的上下文菜单

下一步，我们采用菜单编辑器创建上下文菜单。这个菜单以 uicontextmenu 对象开始，带有四个选项（如图 10.30c），同样，我们设置各项的 Label，Tag 和 callback 字符串。

到这里，我们把 GUI 保存为 plot_line.fig，plot_line.m 将被自动创建。然而，每项哑元回叫函数不会被自动创建，必须手工完成。

GUI 创建后，必须为菜单建立六个回叫函数。最难的回叫函数是 File/Open 菜单项的函数，这个回叫函数必须提示用户输入文件名（使用 uigetfile 对话框），打开文件，读取数据，把它保存到 x 和 y 数组中，最后关闭文件。接着，绘制线条，以应用程序数据保存线条句柄以便后面我们可以修改线条风格。最后，回叫函数必须把上下文菜单联结在一起。mFileOpen_Callback 函数如图 10.31 所示。注意函数使用对话框告知用户文件打开出错。

剩下的回叫函数相当简单，mFileExit_Callback 函数仅仅是关闭图形，线条风格函数仅仅设置线条风格。当用户鼠标在线条上面右击时，上下文菜单会弹出。如果用户从弹出的菜单选择一项，相应的回叫函数将使用保存的线条句柄更改属性。这五个函数如图 10.32 所示。

程序最终如图 10.33。在你自己的电脑上实践，确保它正确运行。

```
% ----------------------------------------------------------------
function mFileOpen_Callback(hObject, eventdata, handles)
% Get the file to open
[filename, pathname] = uigetfile('*.dat','Load Data'); %%%%获取文件名
if filename ~= 0
    % Open the input file
    filename = [pathname filename];
    [fid, msg] = fopen(filename, 'rt'); %%%%打开文件
    % Check to see of the open failed.
    if fid < 0
        % There was an error -- tell user.
        str = ['File' filename ' could not be opened.'];
        title = 'File Open Failed';
        errordlg(str, title,'modal'); %%%%打开文件失败时的错误信息
    else
        % File opened successfully, Read the (x,y) pairs from
        % the input file. Get first (x,y) pair before the
        % loop starts.
        [in, count] = fscanf(fid, '%g', 2); %%%%读取数据
        ii = 0;
        while ~feof(fid)
            ii = ii + 1;
            x(ii) = in(1);
            y(ii) = in(2);
```

```
            %Get next (x,y) pair
            [in, count] = fscanf(fid, '%g', 2);%%%%读取数据
        end
        % Data read in. Close file.
        fclose(fid);
        % Now plot the data.
        hline = plot(x,y,'LineWidth',3); %%%%绘线条
        xlabel('x');
        ylabel('y');
        grid on;
        % Associate the context menu with line
        set(hline,'Uicontextmenu', handles.ContextMenu1); %%%%设置上下文菜单
        %Save the line's handle as application data
        handles.hline = hline;    %%%%把句柄保存为应用程序数据
        guidata(gcbf, handles);
    end
end
```

图 10.31 File/Open 回叫函数

```
% ------------------------------------------------------------------
function mFileExit_Callback(hObject, eventdata, handles)
close(gcbf);
% hObject      handle to mFileExit (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)


function LineSolid_Callback(hObject,eventdata,handles)
set(handles.hline,'LineStyle','-');

function LineDashed_Callback(hObject,eventdata,handles)
set(handles.hline,'LineStyle','--');

function LineDotted_Callback(hObject,eventdata,handles)
set(handles.hline,'LineStyle',':');

function LineDashDot_Callback(hObject,eventdata,handles)
set(handles.hline,'LineStyle','-.');
% ------------------------------------------------------------------
```

图 10.32 plot_line 中的其它函数

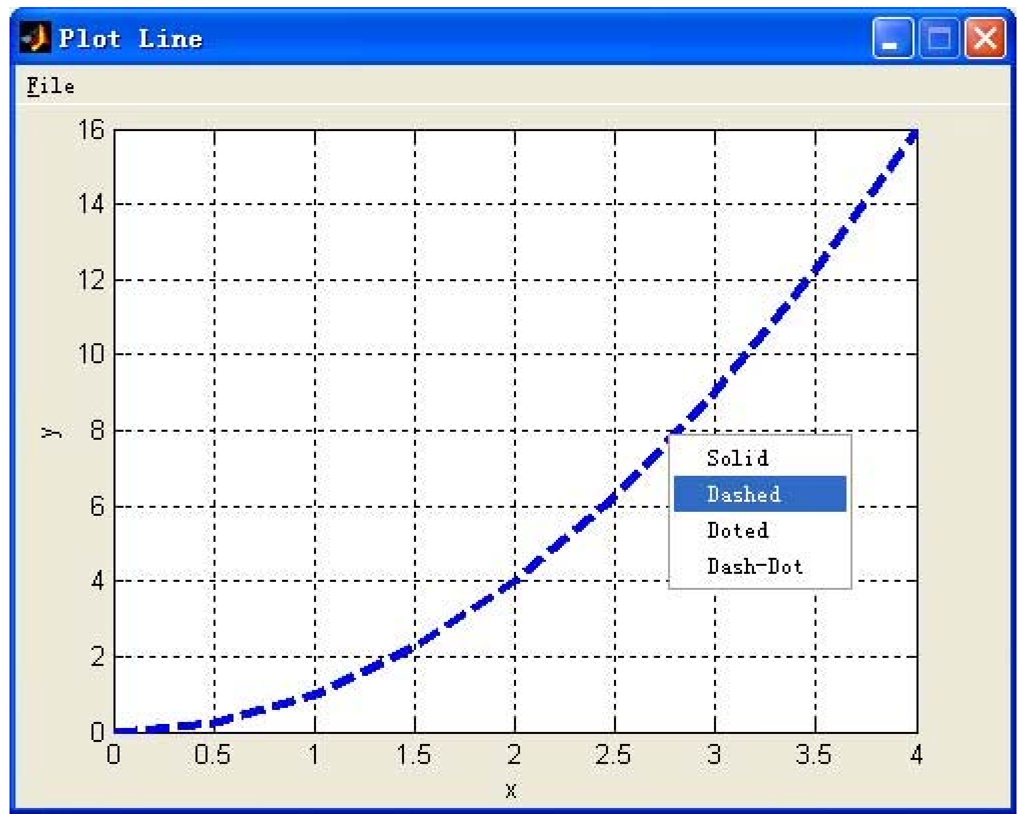


图 10.33 plot_line 产生的 GUI

测试 10.1

本测试检查你是否已经明白了 10.1 节到 10.6 节介绍的概念。如果你完成本测试有困难，重读以上章节，问你的老师，与你的同学讨论。可在本书后面找到本测试的答案。

1. 列出本章中介绍到的图形组件的类型。每一个目的是什么？
2. 什么是回叫函数？ **MATLAB** 的 GUIs 是如何使用回叫函数的？
3. 描述创建基于 GUI 程序的步骤。
4. 描述数据结构句柄的目的。
5. **MATLAB** GUI 中是如何保存应用程序数据的？为什么要保存 GUI 中的应用程序数据？
6. 如何使图对象不可见？如何关闭图对象对鼠标的单击或键盘输入的响应？
7. 本章中哪些 GUI 组件能对鼠标作业响应？哪些能对键盘输入作出响应？
8. 什么是对话框？如何创建对话框？
9. 模式对话框与无模式对话框之间的区别是什么？
10. 标准菜单与上下文菜单之间的区别是什么？
11. 什么是加速键？什么是助记键？

10.7 创建高效 GUIs 的技巧

本节列出一些创建高效图形用户界面（GUI）的技巧。

10.7.1 工具提示

MATLAB 从 5.2 版本开始支持工具提示——即当鼠标在某对象上面停留一小会儿后弹出的一个小帮助窗口，它是 uicontrol GUI 对象中的一个属性，通常用来对 GUI 上的对象提供快速帮助。

工具提示通过在对象的 TooltipString 属性中设置你想要显示的内容来定义。在本章末练习中，你会被要求创建工具提示。

好的编程习惯

对 GUI 组件设置工具提示，为用户提供关于该组件功能的有用线索。

10.7.2 伪代码（p 码，pcode）

在程序执行过程中第一次执行函数时，**MATLAB** 把函数编译（或解析）成称为 pcode（伪代码的简称）的中间代码，然后执行它的运行时解析程序中执行伪代码。一旦函数被编译，它就留中 **MATLAB** 的内存中，能够被重复执行而无需重新编译。不过，当 **MATLAB** 下次执行时，函数又得重新编译。

这种初期编译带来的开销（不利因素）相对较小，不过随着函数越来越大，这种开销变得越来越重要。由于定义 GUI 的函数通常相当大，基于 GUI 的程序编译开销与其它类型的程序相比也相对较大。换句话说，由于初期编译，GUI 程序运行得更慢。

幸运的是，我们可以避免这种开销：把 **MATLAB** 函数及脚本文件编译成伪代码，保存的伪代码文件可以在将来立即执行。执行伪代码文件节省了初期编译时间，使程序运行得更快。

MATLAB 采用 pcode 命令创建伪代码文件，这个命令采用下面的形式之一：

```
pcode fun1.m fun2.m fun3.m ...
pcode *.m
```

第一种形式编译给定名称的文件，第二种形式编译当前目录下所有的 M 文件。编译结果以“p”保存。例如，你编译了文件 foo.m，那么输出将保存在 foo.p 文件中。

如果同一函数既存在于 M 文件中也存在于 p 文件中，**MATLAB** 将自动执行 p 文件中的版本，这是由于该版本更快。然而，如果你修改了 M 文件，你一定要记得重新编译，否则程序将仍然执行旧代码。

把文件编译成伪代码也有其它优点。在伪代码的形式把发布给其他人可以保护你在源代码上的投资。它们可以自由执行，但别人就没那么容易重建文件得到你的（设计）理念。

好的编程习惯

一旦程序工作正常，用 pcode 命令预编译 M 文件，以便提高程序运行速度。

编程隐患

如果更改了已经编译成伪代码的 M 文件，记得要重新编译。否则，你仍然是那些老的、没有修改的代码。

10.7.3 附加提高

基于 GUI 的程序可能比我们在本章所简单介绍的更复杂。作为本章使用过的 Callback 属性的附加内容，uicontrols 支持三种类型的回叫：CreateFcn，DeleteFcn 和 ButtonDownFcn。

MATLAB 图形同样支持三种重要类型的回叫：WindowButtonDownFcn，WindowButtonMotionFcn 和 WindowButtonUpFcn。

CreateFcn 属性定义了对象创建时自动调用的回叫。这种属性允许程序员在程序运行期间对象创建时自定义该对象。由于这种回叫在对象完成定义之前执行，程序员必须在对象创建之前指定描述对象根属性的函数。例如，下面的语句将会使得 function_name 函数在每次 uicontrol 对象创建之前执行。函数将会在 **MATLAB** 创建对象的属性之后被调用，因此它们（对象的属性）在函数执行之前都是可用的。

```
Sset(0,'DefaultUicontrolCreateFcn', 'function_name')
```

DeleteFcn 属性定义了对象被销毁时自动调用的函数，它在对象的属性被销毁之前执行，所以它们（对象的属性）在函数执行之前仍可用。这个回叫使程序员有机会去做一些自定义的清理工作。

ButtonDownFcn 属性定义了当鼠标在 uicontrol 周围 5-pixel 内按按钮时自动调用的回叫。如果鼠标按钮按在 uicontrol 上，执行 Callback 函数，否则，如果是在边界附近，则执行 ButtonDownFcn。如果 uicontrol 不可用，此时即使鼠标击在控件上，也会执行 ButtonDownFcn。

WindowButtonDownFcn、**WindowButtonMotionFcn** 和 **WindowButtonUpFcn** 等图形级的回叫函数允许程序员实现动画或拖放等特性，这是由于这些函数可以检测鼠标按下后的初始、中间和最终位置。这些函数跳出本书讨论的范围，但是值得学习知道。参阅 **MATLAB** 用户文档《创建图形用户界面》一卷获得这些回叫函数的描述。

例 10.3

创建柱状图

编写程序，打开用户指定的数据文件，按文件中的数据计算柱状图。程序应该能计算文件中数据的平均差，中位差和标准差。程序必须包含文件菜单（含打开和退出项），还必须提供用户更改图形柱数的方法。

选择另一种颜色替换图形和文本标签的背景色，为菜单项设置键盘加速键，在适当的地方添加工具提示。

解决方案：

本程序应包含带有打开和退出项的标准菜单、一个用来绘制柱状图的坐标轴，六个文本域，其中三个用来显示平均差、中位差和标准差，三个用来做为标签。程序还必须包含一个标签和一个编辑框，用来让用户更改柱状图中的柱数。

我们选取淡蓝色[0.6 1.0 1.0]作为 GUI 的背景色，为了达到目的，应在设计时在属性编辑器中把这个代表颜色的向量加进图形的“Color”属性和文本标签的“BackgroundColor”属性。

第一步，使用向导 guide 设计 GUI（如图 10.34a 所示），然后，用属性编辑器编辑界面中的文本域和编辑框。必须为这些控件设置独一的名称，以便在后面的回叫函数中定位。下一步，使用菜单编辑器创建文件菜单（如图 10.34b）。最后，把 GUI 保存为 histGUI，向导会自动创建 histGUI.fig 和 histGUI.m。

histGUI.m 保存后，必须编辑它，在句柄结构中添进表示柱状图的柱数的应用程序数

```
% Choose default command line output for histGUI
handles.output = hObject;

% Add the number of bins to this structrue.
handles.nbins = str2num(get(handles.NBins,'String'));

% Update handles structure
guidata(hObject, handles);
```

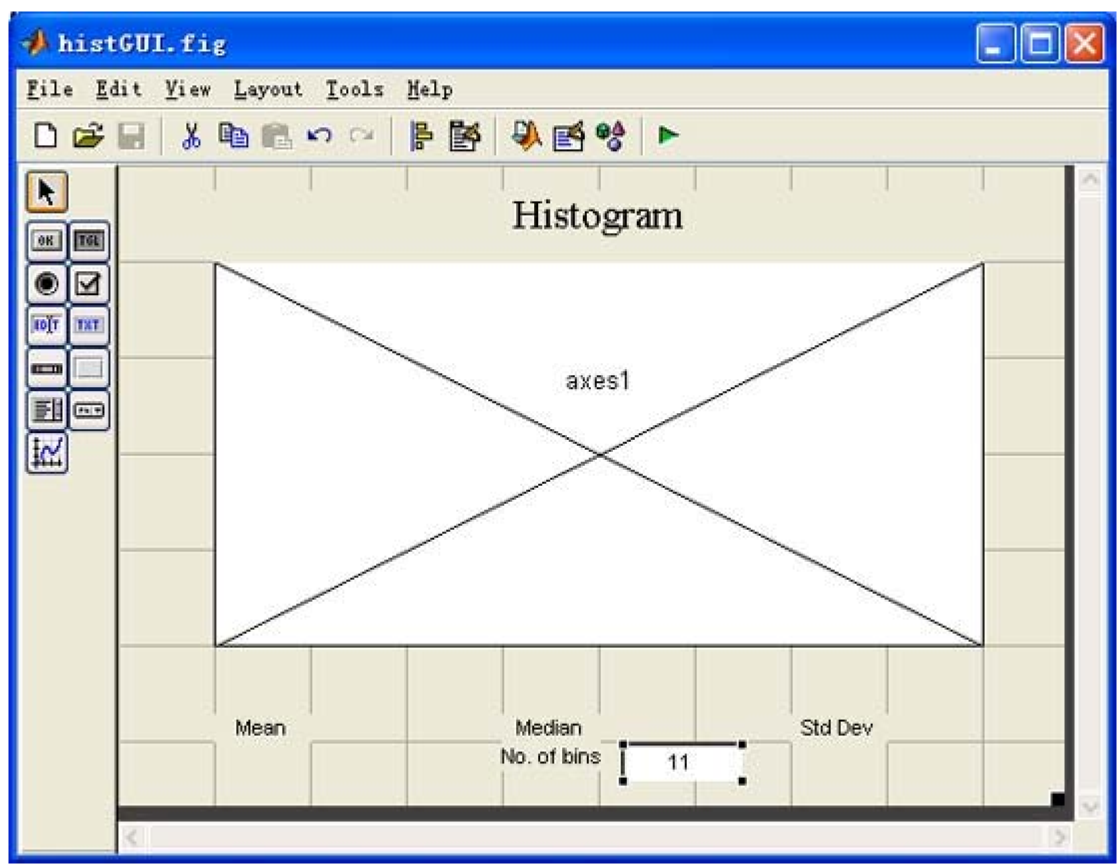


图 10.34 （a）histGUI 的布局

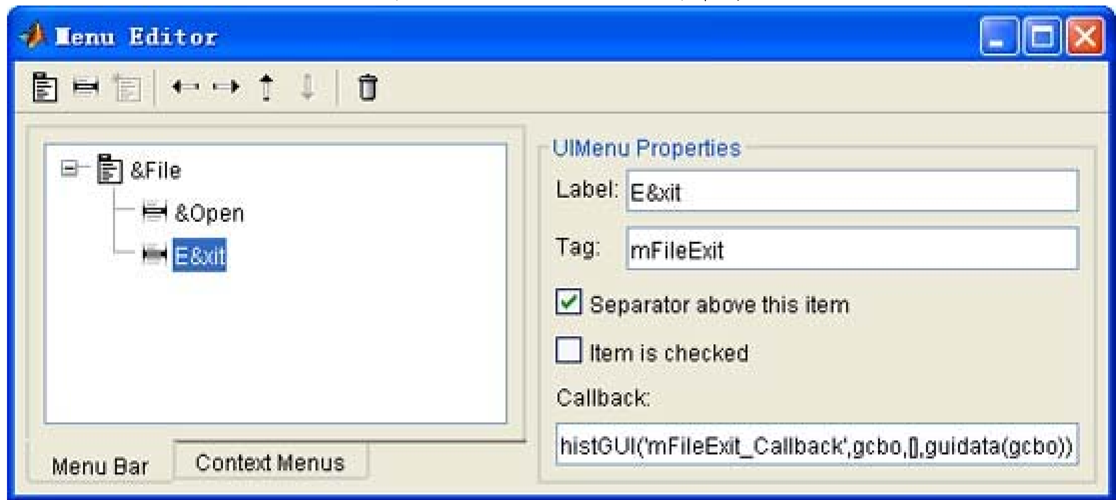


图 10.34 （b）菜单编辑器中的文件菜单

接着，为“文件/打开”、“文件/退出”菜单项和“柱数”编辑框创建回叫函数。只有最后一个函数才会自动创建——菜单项的回叫必须手工添加。

“文件/打开”回叫函数提示用户选择文件，然后从文件中读取数据，计算并显示柱状图，更新统计文本域。注意文件中的数据也必须保存到 handles 结构中，以便用户更改柱数时可以重新计算。处理这些的步骤的函数如下：

```
function mFileOpen_Callback(hObject,eventdata, handles)
% Get file name
[filename, path] = uigetfile('*.dat;*.abc', 'Load Data File');
if filename ~= 0

    % Read data
```

```
        x = textread([path filename],'%f');

        % Save in handles structure
        handles.x = x;
        guidata(gcbf, handles);
        % Create histogram
        hist(handles.x, handles.nbins);

        % Set axis labels
        xlabel('\bfValue');
        ylabel('\bfCount');

        % Calculate statistics
        ave = mean(x);
        med = median(x);
        sd = std(x);
        n = length(x);

        % Update fields
        set (handles.MeanData,'String',sprintf('%7.2f',ave));
        set (handles.MedianData,'String', sprintf('%7.2f',med));
        set (handles.StdDevData,'String', sprintf('%7.2f',sd));
        set (handles.TitleString,'String',['Histogram (N = ' int2str(n) ')']);
    end
```

“文件/退出”很简单，仅是关闭图形而已。

```
function mFileExit_Callback(hObject, evendata, handles)
close(gcbf);
```

NBins 的回叫函数必须读取输入的数字，取最接近的整数，然后又把它显示在编辑框中，并重新计算和显示柱状图。注意柱数必须重新保存到 handles 结构中，以便用户打开新的数据文件时可以使用。实现这些步骤的回叫函数如下：

```
function NBins_Callback(hObject, eventdata, handles)
% Get number of bins, round to integer,and update field
nbins = str2num(get(gcbo,'String'));
nbins = round(nbins);
if nbins < 1
    nbins = 1;
end
set (handles.NBins, 'String', int2str(nbins));
% sprintf('nbins = %d' nbins);

% Save in handles structrue
handles.nbins = nbins;
guidata(gcbf, handles);

% Re-display data, if available
if handles.nbins > 0 & ~isempty(handles.x)
    % Create histogram
    hist(handles.x, handles.nbins);
end
```

程序最终如图 10.35 所示，试验该程序能否正确运行。

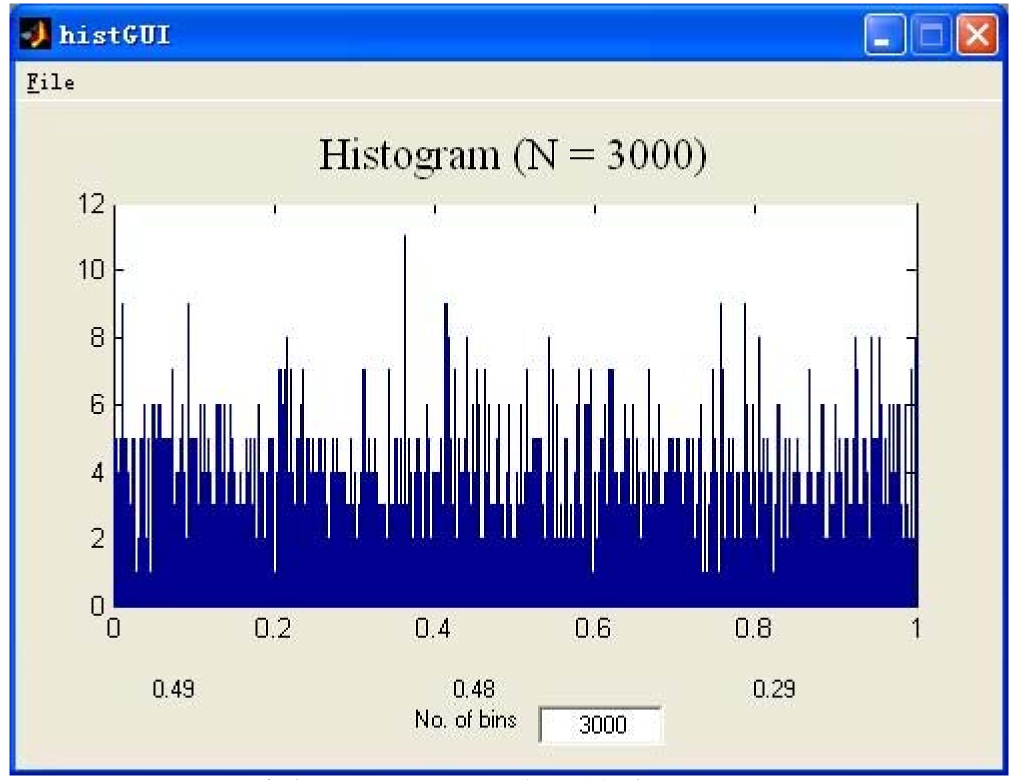


图 10.35 histGUI 程序产生的 GUI（数据采用随机产生的 3000 个 0-1 之间的数）

10.8 总结

本章我们学习了如何创建 **MATLAB** 图形用户界面，一个 GUI 的三个基本部分是组件（uicontrols，uimenus 和 uicontextmenus），包含它们的图形和以鼠标及键盘作出响应实现动作的回叫。

标准 GUI 组件由 uicontrol 创建，它包括文本域，编辑框，框架，按钮，形状按钮，复选框，单选按钮，下拉菜单，列表框和滑动条。由 uimenu 和 uicontextmenu 创建的标准 GUI 组件是标准菜单和上下文菜单。

所有的这些组件都可以使用向导 guide（GUI 开发环境工具）放置到图形上。一旦 GUI 布局完成，用户必须用属性编辑器编辑对象属性，然后为每个 GUI 对象编写实现其动作的回叫函数。

对话框是用来显示信息或从用户处获取输入的特殊图形。对话框通常用来显示错误，提供警告，询问问题或获取用户输入。它们也用来选择文件与打印机属性。

对话框有有模式与无模式之分。模式对话框在它消失之前不允许用户访问应用程序的其它窗口，而无模式对话框没有这个限制。模式对话框习惯上用来显示需要用户急切注意、不可忽略的警告或错误信息。

菜单也可以添加到 **MATLAB** 的 GUI 中。菜单可以使界面没有太多组件又可以实现动作。菜单在处理那些不会经常用到的选项而又不想把 GUI 堆得乱七八糟时非常有用。菜单采用菜单编辑器来创建，对每个菜单项，用户必须使用菜单编辑器设置其 Label 标签、Tag 名称和回叫字符串。与 GUI 组件不同，guide 不会自动为菜单项创建回叫字符串，用户要完全负责定义回叫字符串和实现其函数体。

加速键与键盘助记键可以用来加快窗体操作。

10.8.1 好的编程习惯总结

当使用 **MATLAB** 的 GUI 时，应该遵循下面的指导语：

1. 用 guide 对一个新的用户图形界面进行布局，并用属性编辑器对每一个组件的初始属性进行设置，例如显示在组件上的文本，组件的颜色，还有回叫函数的名字。
2. 用 guide 创建完一个用户图形界面后，人工编辑产生的函数，增加注释，描述这个函数的目的和组件，执行回叫函数的代码。
3. 把GUI应用程序数据存储到handles结构中，以便任意的一个回叫函数都可以应用它。
4. 如果你修改了 handles 结构中的任何 GUI 应用数据，确保在函数退出之前保存了调用 guidata 的结构。
5. 在基于 GUI 编程中，使用对话框来提供信息或要求输入数据，如果信息紧迫且不可忽略，则把对话框设为模式对话框。
6. 对 GUI 组件设置工具提示，为用户提供关于该组件功能的有用线索。
7. 一旦程序工作正常，用 pcode 命令预编译 M 文件，以便提高程序运行速度。

10.8.2 MATLAB 总结

下面的总结列出了本章中讨论到的 **MATLAB** 命令与函数，还有它们的简要介绍。同时，请参阅表 10.2，10.3，10.5 和 10.6 等图形对象属性总结。

命令与函数

axes	创建一组坐标轴的函数
dialog	创建一个通用对话框
errordlg	显示错误信息
helpdlg	显示帮助信息
findobj	查找一个或多个属性匹配的 GUI 对象
gcbf	取得回叫图形
gcbo	取得回叫对象
guidata	保存图形中的 GUI 应用程序数据
guihandles	从保存在图形中的应用程序数据中取得 handles 结构
guide	GUI 开发环境工具
inputdlg	从用户处获取输入数据的对话框
printdlg	打印对话框
questdlg	询问问题的对话框
uicontrol	创建 GUI 对象的函数
uicontextmenu	创建上下文菜单的函数
uigetfile	获取输入文件的对话框
uimenu	创建标准菜单、或者在标准菜单或上下文菜单中创建菜单项的函数
uiputfile	选择输出文件对话框
uisetcolor	显示颜色选择对话框
uisetfont	显示字体选择对话框
warndlg	显示警告对话框

10.9 练习

10.1 描述 **MATLAB** 中创建 GUI 所需的步骤。

10.2 回叫函数如何工作？回叫函数如何定位其所要操作的对象或图形。

10.3 创建一个 GUI 程序，使用下拉菜单选择 GUI 的背景颜色。

10.4 创建一个 GUI 程序，使用标准菜单选择 GUI 的背景颜色。

10.5 写一个 GUI 程序，绘制函数 $y(x) = ax^2 + bx + c$ 的图象。程序应该包含一组坐标轴，包含用来输入 a，b，c，x 的最大值和最小值的 GUI 元素，还必须有 GUI 元素的工具提示。

10.6 修改 10.5 的 GUI，加入菜单，菜单中包含两个子菜单，一个用来选择图象线条的颜色，一个用来选择图象线条的风格。必须有选择标记显示所选择的项。同时，还应该有“退出”项，当用户选择退出时，弹出一个模式询问对话框，显示“Are You Sure?”以便进一步确认。

10.7 修改 10.4.8 节中的列表框例子，允许对列表框中的项多选，当点击“Selected”按钮时，文本域中显示列表框中所有选择的项。

10.8 **随机数分布** 创建一个 GUI，显示不同的类型的随机数分布。程序应该能够产生 200,000 个随机数并用 hist 创建柱状图显示出来。注意正确设置标题和坐标轴的标签。

程序应能支持统一、高斯和瑞利分布，用下拉菜单来选择分布类型，此外，还必须提供一个编辑框，以便用户可以修改柱状图的柱数。确保用户的输入合法（必须为正整数）。

10.9 修改例 10.1 中的温度转换 GUI，添加一支“温度计”，温度计的红色“液柱”显示了当前摄氏温度值，范围在 0-100ºC 之间。

10.10 修改练习 10.9 中温度转换 GUI，允许用鼠标调节显示的温度。（警告：本练习需要用到本章中没有讨论到的材料，在线参阅 figure 图形对象的 CurrentPoint 属性。）

10.11 **最小二乘方拟合** 创建一个 GUI，从文件中读取数据对，对数据进行最小二乘方拟合。数据以（x，y）格式储存在磁盘文件中，每行一组数据。用 **MATLAB** 的 ployfit 函数绘制最小二乘方拟合图线，同时还绘制原始数据图线。包含两个菜单：“文件”和“编辑”，文件菜单含有“打开”和“退出”两项，在退出程序之间用户必须收到“Are You Sure?”信息。编辑菜单允许用户自定义显示，包括线条风格、线条颜色和网格显示状况。

10.12 修改练习 10.11，在编辑菜单添加“首选项”，允许用户取消退出时提示“Are You Sure?”。

10.13 修改练习 10.12，允许程序读写初始化文件。该文件应包含程序前一次运行时的线条风格，线条颜色，网格选择（on/off）和退出时的提示设置。这些设置应该在用户从“退出”菜单中退出程序时自动保存，下次程序再运行时被自动加载读取。

附录 A ASCII 字符集

MATLAB 字符串使用 ASCII 字符集，如下表所示，它包括了 127 字符。**MATLAB** 字符串比较依赖于被比较字符在字典中的相对位置。例如，'a'在 ASCII 字符集表中的位置为 97，而'A'的位置为 65，由于 97>65，所以'a'>'A'将返回 1(true)。

每个 **MATLAB** 字符都是储存在一个 16 位域中，这意味着将来 **MATLAB** 能支持整个 Unicode 字符集。不过目前 **MATLAB6** 仍然不能支持 Unicode 字符。

下面的表格包含了 ASCII 字符集，其中头两个数字由行来定义，而第三个数字由列定义。如下，'R'是在 8 行 2 列，因此 82 是它在 ASCII 中的位置。

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl(lf)	vt	ff	cr	so	si	dle(sle)	dc1(cs1)	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub(sib)	esc	fs	gs
3	rs	us	(space)	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

附录 B 测试答案

本附录包括了本书中所有练习的答案

测试 1.1

1.
MATLAB 命令窗口是 **MATLAB** 起动时第一个看到的窗口，用户可以在命令窗口提示符">>"后面输入命令，这些命令会被立即执行。命令窗口也可以用来执行 M 文件。编辑/调试窗口是用来新建，修改或调试 M 文件的。图像窗口用来显示 **MATLAB** 的图形输出。
2.
在 **MATLAB** 中你可以使用下列几种方式获取帮助。

- 在命令窗口中输入 help <command_name>，本命令将会在命令窗口中显示关于些命令的有信息。
- 在命令窗口中输入 lookfor <keyword>，本命令将会在命令窗口中显示所有的在第一注释行中包含该关键字的命令和函数。
- 通过在命令窗口输入 helpwin 或 helpdesk 启动帮助浏览器，或者是在启动板中选择"Help"。帮助浏览器包含了基于超文本的 **MATLAB** 所有特性的描述，HTML 或 PDF 格式的在线手册，这是 **MATLAB** 最全面的帮助资源。

3.
工作区是命令、M 文件或函数执行时被 **MATLAB** 使用的变量或数组的收集器，所有命令都在命令窗口（所有的脚本文件也是从命令窗口执行）共享公共工作区，因此它们也共享所有变量，工作区的内容可以通过 whos 命令来查看，或者通过工作区浏览器来图形化地查看。
4.
要清除工作区的内容，只需在命令窗口中输入 clear 或 clear variables 即可。
5.
执行此操作的命令如下：

```
>> t = 5;
>> x0 = 10;
>> v0 = 15;
>> a = -9.81;
>> x = x0 + v0 * t + 1/2 * a * t^2
x = -37.6250
```

6.
执行此操作的命令如下：

```
>> x = 3;
>> y = 4;
>> res = x^2 * y^3 / (x - y)^2
res =
     576
```

问题 7 或 8 没有单一的“正确”答案。

测试 2.1

1.
数组是在内存中被组织成行和列的数据集合，只有一个名称，数据要通过在数组名后面圆括号里加上表示数据所在行和列的数字来访问。术语"向量"通常用来描述只有一维的数组，而"矩阵"通常用来描述二维或更多维的数组。
2.
(a) 这是一个 3×4 数组；
(b) c(2,3) = -0.6；
(c) 数组中值为 0.6 的元素是 c(1,4)，c(2,1)和 c(3,2)。
3.
(a) 1×3；(b) 3×1；(c) 3×3；(d) 3×2；(e) 3×3；(f) 4×3；(g) 4×1
4.
w(2,1) = 2
5.
x(2,1) = -20i
6.
y(2,1) = 0
7.
v(3) = 3

测试 2.2

1.
(a) c(2,:) = [0.6 1.1 -0.6 3.1]
(b) d(:,4) = $\begin{bmatrix} 0.6 \\ 3.1 \\ 0.0 \end{bmatrix}$
(c) c(1:2,2:4) = $\begin{bmatrix} -3.2 & 3.4 & 0.6 \\ 1.1 & -0.6 & 3.1 \end{bmatrix}$
(d) c(6) = 0.6
(e) c(4,end) = [-3.2 1.1 0.6 3.4 -0.6 5.5 0.6 3.1 0.0]
(f) c(1:2,2:end) = $\begin{bmatrix} -3.2 & 3.4 & 0.6 \\ 1.1 & -0.6 & 3.1 \end{bmatrix}$
(g) c([1 3],2) = $\begin{bmatrix} -3.2 \\ 0.6 \end{bmatrix}$
(h) c([2 2],[3 3]) = $\begin{bmatrix} -0.6 & -0.6 \\ -0.6 & -0.6 \end{bmatrix}$
2.
(a) a = $\begin{bmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix}$ (b) a = $\begin{bmatrix} 4 & 5 & 6 \\ 4 & 5 & 6 \\ 4 & 5 & 6 \end{bmatrix}$ (c) a = $\begin{bmatrix} 4 & 5 & 6 \\ 4 & 5 & 6 \end{bmatrix}$
3.
(a) a = $\begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 3 \\ 0 & 0 & 1 \end{bmatrix}$ (b) a = $\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 6 \end{bmatrix}$ (b) a = $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 9 & 7 & 8 \end{bmatrix}$

测试 2.3

1.

要求的命令为“format long e”。

2.

(a) 这些语句读取用户的圆的半径，然后计算并显示圆的面积。

(b) 这些语句用整数显示 π 值，所以显示的字符串为“The value is 3!”。

3.

第一个语句采用指数形式输出 12345.67，第二个语句采用浮点数格式输出该值，第三个语句以一般形式输出该值，第四个语句采用 12 位字符宽，并且小数点后四位的形式输出。这些语句的结果如下：

```
value = 1.234567e+004
value = 12345.670000
value = 12345.7
value =      12345.6700
```

测试 2.4

1.

(a) 操作非法。数据相乘必须有相同形式，或者数组与标量之间相乘。

(b) 合法矩阵相乘：result = $\begin{bmatrix} 4 & 4 \\ 3 & 3 \end{bmatrix}$

(c) 合法数据相乘：result = $\begin{bmatrix} 2 & 1 \\ -2 & 4 \end{bmatrix}$

(d) 操作非法。矩阵相乘 b * c 产生一个 1×2 数组，而 a 是一个 2×2 数组，故相加非法。

(e) 操作非法。在两个不同大小的数组之间相乘 b .* c 是非法的。

2.

结果可通过 x = A/B 操作得到：x = $\begin{bmatrix} -0.5 \\ 1.0 \\ -0.5 \end{bmatrix}$

测试 3.1

1. a > b	1
2. b > d	0
3. a > b & c > d	0
4 a == b	0
5. a & b > c	0
6. ~~b	1
7. a & b > c	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$
8. a > c & b > c	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

9. 非法。因为两个数组大小不一样。

10. a*b^2 > a*c	0
11. d | b > a	1
12. (d | b) > a	0
13. isinf(a/b)	0
14. isinf(a/c)	1
15 a > b & ischar(d)	1
16. isempty(c)	0

测试 3.2

1.

```
if x >=0
   sqrt_x = sqrt(x);
else
   disp('ERROR: x < 0');
end
```

2.

```
if abs(denominator) < 1.0E-300
   disp('Divide by 0 error.');
else
   fun = numerator / denominator;
   disp(fun);
end
```

3.

```
if distance <= 100
   cost = 0.50 * distance;
elseif distance <= 300
   cost = 50 + 0.30 * (distance - 100);
else
   cost = 110 + 0.20 * (distance - 300);
end
```

4.
这些语句不对。要正常工作，第二个 if 语句必须是 elseif。
5.
这些语句合法，它们将显示“Prepare to stop.”。
6.
这些语句能够执行，但它们不会按程序员的意愿工作。如果温度为 150，这些语句将输出“Human body temperature exceeded”，而不是“Boiling point of water exceeded”，这是因为 if 结构执行了第一个为 true 的语句而跳过其它的。要正确工作，这些测试条件的顺序要反过来。

测试 3.3

1.

```
x = 0:pi/10:2*pi;
x1 = cos(2*x);
y1 = sin(x);
plot(x1,y1,'-ro','LineWidth',2.0,'MarkerSize',6,'MarkerEdgeColor','b','MarkerFaceColor','b')
```

3.

```
'\itf\rm(\itx\rm) = sin \theta cos 2\phi'
```

4.

```
'\bfPlot of \Sigma \itx\rm\bf^{2} versus \itx'
```

5.
这个字符串创建字符：$\boldsymbol{\tau}_m$
6.
这个字符串创建字符：$x_1^2 + x_2^2$(units:$\mathbf{m}^2$)
7.
显示反斜线采用双反斜线('\\')

测试 4.1

1.
4 次
2.
0 次
3.
1 次
4.
2 次
5.
2 次
6.
ires = 10
7.
ires = 55
8.
ires = 25;
9.
ires = 49;
10
使用循环与分支：

```
for ii = -6*pi:pi/10:6*pi
    if sin(ii) > 0
        res(ii) = sin(ii);
    else
        res(ii) = 0;
end
```

使用向量代码：

```
arr1 = sin(-6*pi:pi/10:6*pi);
res = zeros(size(arr1));
res(arr1>0) = arr1(arr1>0);
```

测试 5.1

1.
脚本文件是保存在文件中的 **MATLAB** 语句集合。脚本文件共享命令窗口工作区，所以任何之前运行的脚本文件中定义的变量都能被当前脚本文件使用，脚本文件定义的变量在文件执行之后还保留在工作区中。脚本文件没有输入参数，也没有输出参数，不过脚本文件之间可以通过工作区交换数据。相反，每一个 **MATLAB** 函数运行在它自己的独立工作区，函数通过输入参数列表获取输入数据，通过输出参数列表给调用者返回数据。
2.
help 命令显示某个函数的所有注释行，直到遇到空白注释行或执行语句为止。
3.
H1 注释行是文件中注释的第一行，这一行可以被 lookfor 命令搜索并显示。应该把该函数的用途摘要写在这一行中。
4.
在值传递机制中，每个输入参数的副本而不是参数本身从调用者传递给函数。这样设计可以避免输入数据在函数内被自由的修改，这可能并不是调用者实际上需要的，因而也为设计好程序提供保证。

5.
MATLAB 函数可以有任意数量的参数，并且并不是每次函数调用时，每个输出参数都必须具备。nargin 函数用来确定函数被调用时实际提供了多少个参数，而 nargout 函数用来确定函数被调用后实际上有多少个输出参数。
6.
函数调用不正确。调用 test1 必须提供二个输入参数。在这种情况下，变量 y 在 test1 中将没有定义，函数被忽略。
7.
函数调用正确。

测试 6.1

1.
(a) result = 1，因为比较的是数的实部。
(b) result = 0，因为两个数的绝对值是确定的。
(c) result = 25
2.
plot(array)函数绘制了数组的虚部和实部，实部在 x 轴上，而虚部在 y 轴上。
3.
该向量可以通过 double 函数转换。
4.
这些语句把两行连结在一起，变量 res 包含字符串“This is a test! This line, too.”。
5.
这些语句非法——没有 strcati 函数。
6.
这些语句非法——这两个字符串必须有相同的列数，可这些字符串长度不同。
7.
这些语句合法，产生结果

$$res = \begin{bmatrix} \text{This is a text!} \\ \text{This line, too.} \end{bmatrix}$$

注意现在每行有 17 个字符，第二行垫长了长度。
8.
这些语句合法，结果是 res = 1，这是由于这两个字符串前 5 个字符相同。
9.
这些语句合法，结果是 res = [4 7 13]，这是由于字母“s”在字符串中的那些位置出现。
10.
这些语句合法，结果是 res = 'This IS a test!'。
11.
这些语句非法——使用 strncmp 时你必须指定要比较的字符个数。

测试 7.1

1.
稀疏数组(sparse array)是一种只为数组中的非零元素分配内存的特殊类型数组，内存中存储了稀疏数组中非零元素的下标和值。相反地，不管元素的值是不是 0，都为每个元素分配内存称为全数组。稀疏数组可以通过用 full 函数转换成全数组，全数组也可以通过用 sparse 函数转换成稀疏数组。
2.
胞数组(cell array)一个“指针”数组，每个元素都可以指向任何类型的 **MATLAB** 数据，它们可以是胞数组中的普通数组，也可以是不同类型的数据，例如一个数字数组，一个字符串

或者是另一个胞数组，一个结构等等。此外，胞数组使用花括号{}而不是圆括号()来选择与显示单元中的内容。
3.
内容索引(Content indexing)包括在单元下标前后放置{}和以普通方式标记单元内容，这种索引方式定义了胞中数据结构的内容。
胞索引(Cell indexing)包括在保存到胞中的数据前后放置{}和以普通下标记号标记的胞下标。这种索引方式创建了包含指定数据的数据结构，并把结构赋给胞。
4.
结构是每个独立元素都有一个名称的一种数据类型，结构里的每个元素称为域，一个结构中的域可以有不同类型，域的定位是通过在结构名后附上域名的，结构名与域名之间用逗号分隔开。结构可以是普通数组或普通数组中的胞数组。结构元素通过名称来定位。
5.
varargin 函数出现在输入参数的最后一项，它返回一个胞数组，数组包含了函数被调用时实际上指定的参数，每一个参数保存在胞数据中独立元素中。这个函数允许 **MATLAB** 函数支持任意数量的输入参数。
6.
(a) a(1,1) = (3x2 double).胞数据元素 a(1,1)的内容是一个 3x3 双精度数组，这种数据结构已经显示出来。
(b) a(1,1) = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$，这个语句显示了保存在元素 a(1,1)中数据结构的值。
(c) 这些语句非法，这是由于你不能把一个数据结构乘以一个值。
(d) 这些语句合法，这是由于你可以把一个数据结构的内容乘以一个值。结果是
$\begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$
(e) a{2,2} = $\begin{bmatrix} -4 & -3 & -2 \\ -1 & 0 & 1 \\ 2 & 3 & 4 \end{bmatrix}$
(f) 这个语句合法。用$\begin{bmatrix} -17 \\ 17 \end{bmatrix}$这样一个 2×1 双精度数组初始化胞数组元素 a(2,3)。
(g) a{2,2}(2,2) = 0。
7.
(a) b(1).a - b(2).a = $\begin{bmatrix} -3 & 1 & -1 \\ -2 & 0 & -2 \\ -3 & 3 & 5 \end{bmatrix}$。
(b) strncmp(b(1).b,b(2).b,6) = 1，这是由于两个结构元素的包含字符串元素的前 6 个字符相同。
(c) mean(b(1).c) = 2
(d) 这个语句非法，这是由于你不能把结构数组的个别元素当作数组本身。
(e) b = 1x2 struct array with fields:

```
            a
            b
            c
```

(f) b(1) =

```
    a:[3x3 double]
    b:'Element 1'
    c:[1 2 3]
```

测试 8.1

1.
textread 函数是设计用来读取格式化好的 ASCII 文件到数据列中，每一列的格式可以是不同类型。这个命令在导入由其它程序输出的表格数据时相当有用，这是因为它可以处理单一文

件中的不同类型的数据。
2.
MAT 文件相对有效地使用磁盘空间，它保存了变量的全精度——没有精度损失，如保存为 ASCII 格式，则在转换过程中会有精度损失。此外，MAT 文件保留了工作区中每个变量的全部信息：包括类别，名称，以及是不是全局的。缺点是 MAT 文件为 **MATLAB** 专用，不能与其它程序共享数据。
3.
fopen 函数用来打开文件，fclose 函数用来关闭文件。在 PCs（不是 Unix 计算机）中，文本文件与二进制文件是不同的。如果要以文本方式打开文件，则必须在 fopen 函数中的允许（permission）参数在加上“t”。
4.

```
fid = fopen('myinput.dat','at')
```

5.

```
fid = fopen('input.dat','r');
if fid < 0;
disp('File input.dat does not exist.');
end
```

6.
这些语句不正确，它们以文本方式打开文件，但却以二进制方式读取数据。（应该用 fscanf 函数来读取文本数据，这我们会在后继章节看到。）
7.
这些语句正确。它们创建了一个 10 个元素数组 x，打开一个二进制文件 file1,把数组写入文件，然后关闭它。接着，文件又被打开，并以[2 Inf]的格式把数据读取到数组 array 中去，最终数组的内容是$\begin{bmatrix} 1 & 3 & 5 & 7 & 9 \\ 2 & 4 & 6 & 8 & 10 \end{bmatrix}$。

测试 8.2

1.
格式化 I/O 操作产生格式化文件，一个格式化文件以 ASCII 文本保存可读的字符、数字等等。格式化文件的优点是所保存的数据可以直到看到，并且在不同程序之间方便地交换数据。然而，格式化 I/O 操作花费更多的时间读出和写入，并且格式化文件占用了更多的磁盘空间。非格式化 I/O 操作把内存中的信息不需要经过转换直接复制到磁盘，由于没有转换的过程，所以这些操作比格式化 I/O 操作要快得多。此外，相同的数据占用的磁盘空间也少得多。然而，非格式化数据被人直接检查与翻译。
2.
当要在 **MATLAB** 与其它程序之间交换数据，或者人们要检查与修改文件中的数据时，就应该使用格式化 I/O 操作，否则，就应该使用非格式化 I/O 操作。
3.

```
fprintf('            Table of Cosines and Sines\n\n');
fprintf('   theta            cos(theta)      sin(theta)\n');
fprintf('   ====             =======              ========\n');
for ii = 0:0.1:1
    theta = pi * ii;
    fprintf('%7.4f      %11.5f        %11.5f\n', theta, cos(theta), sin(theta));
end
```

4.
这些语句不正确。变量 a 和 b 可以被正确打印出来，但是%d 格式描述符将没办法正确显示字符串。由于%d 的作用，字符串的第一个字符将显示成十进制数字，而其余的字符在%g 的作用下将被正确显示出来。
5.

从技术上讲，这些语句正确，但结果却不是所预想的。这些语句确实可以在一个文件中混合保存二进制数据和格式化数据，但这之后不管出于何种目的使用这个文件将变得非常困难。通常应该把二进制数据与格式化数据分开放到独立文件中。

测试 10.1

下表列出了在本章中讨论到的图形组件类型及它们的用途。

表 B-1 第十章中讨论到的 GUI 组件

组件	创建者	描述
图形控件		
按钮 Pushbutton	uicontrol	创建一个按钮的图形组件，用鼠标单击时触发回叫信号。
开关按钮 Toggle Button	uicontrol	创建一个开关按钮的图形组件，开关按钮要么是“开（on）”要么是“关（off）”，每次单击时都改变状态，每次鼠标单击时都触发回叫信号。
单选按钮 Radio Button	uicontrol	单选按钮是一种开关按钮，当它被选中时圆圈内显示一个小点。单选按钮组用来实现排它性选择，每次单击鼠标时会触发回叫信号。
复选按钮 Check Box	uicontrol	复选按钮是一种开关按钮，当它被选中时里面显示一个小方块。每次单击鼠标时会触发回叫信号。
编辑框 Edit Box	uicontrol	编辑框用来显示文本，也允许用户修改显示的信息。用户按回车键（Enter）触发回叫信号。
列表框 List Box	uicontrol	列表框用来显示一系列的文本字符串。用户通过单击或双击选择一行文本字符串。当用户选择字符串时触发回叫信号。
弹出菜单 Popup Menus	uicontrol	弹出菜单在响应用户鼠标单击时显示一系列的文本字符串。当没有单击弹出菜单时，只有当前选择的字符串可见。
滑动条 Slider	uicontrol	滑动条一个通过用鼠标拖动该控件平滑、连续地调节数值的图形控件。每次滑动条改变时都触发回叫信号。
静态元素		
框架 Frame	uicontrol	创建的框架是图形窗口中的一个矩形框，框架通常用来把几个控件分为一组。框架不会触发回叫信号。
文本域 Text Field	uicontrol	创建一个标签，在图形窗口中的某处显示一行字符串。文本域不会触发回叫信号。
菜单与轴		
菜单项 Menu Items	uimenu	创建一个菜单项。当鼠标在菜单项上释放时触发回叫信号。
上下文菜单 Context Menus	uicontextmenu	创建一个上下文菜单，即当用户鼠标在某个对象上右击时在该对象上面显示的菜单。
轴 Axes	axes	创建一组新轴以便在上面显示数据。轴从来不会触发回叫信号。

2.
回叫函数即是用户在某个特定的 GUI 组件上发生某个动作（如鼠标单击，键盘输入等等）时执行的函数。它们通常用来当用户在 GUI 组件上单击或打字时执行某个动作。回叫函数在 uicontrol，uimenu 或 uicontextmenu 等组件中用“Callback”属性指定。当创建一个新 GUI 时，uicontrol 的回叫函数自动被向导设置成 xxx_Callback。其中 xxx 是相应 GUI 组件的 Tag 属性的值。菜单回叫函数不会被自动创建，它们必须由程序员手动定义。

3.
创建 **MATLAB** GUI 的基本步骤是：
(1) 确定 GUI 需要什么样的元素，每个元素需要什么样的函数。在纸上精略画出组件的布局。
(2) 使用 **MATLAB** 中 guide 向导（GUI 开发环境）在 figure 图形中放置组件。图形的大小，组件的对齐及组件之间的空间都可以向导中的工具来调整。
(3) 使用 **MATLAB** 中的 Property Inspector 属性检查器（内建在向导中）给每个组件指定名称（即“tag”），并且设置它们的属性，比如颜色、要显示的文本等。
(4) 把 figure 图形保存到文件中。当图形被保存后，它们将在磁盘中产生文件名相同、但扩展名不同的两个文件。fig 文件包含了实际创建的 GUI，M 文件包含的则是图形加载的代码，也保存了每个 GUI 元素的回叫函数原型。
(5) 为每个回叫函数编写代码实现行为。
4.
handles 数据结构一个包含了图形中所有组件句柄的结构。每个元素都有组件的名称，也有组件句柄的值。这个结构传递给每个回叫函数，允许每个回叫函数去访问图形中的组件。
5.
应用数据通过把数据添加到 handles 句柄结构中来实现把数据保存到 GUI 中，当结构被修改后，要使用 guidata 函数来保存结构。由于 handles 句柄结构自动传递给每个回叫函数，添加到结构中的附加数据都可以被 GUI 中的回叫函数使用。（任何一个修改 handles 句柄结构的函数在退出之前都必须确保已经调用了 guidata 函数保存修改了的 handles 句柄结构版本。）
6.
图形对象可以通过设置“visible”的属性值为“off”使其不可见。图形对象可以通过设置“Enable”的属性值为“off”使其对用户的鼠标单击或键盘输入没有响应。
7.
（普通）按钮、开关按钮、单选按钮、复选按钮、列表框、弹出菜单和滑动条都对鼠标单击有响应。编辑框对键盘输入有响应。
8.
对话框是一种特殊的、用来显示信息或者获取用户输入的图形。对话框通常用来显示错误、提出警告、提问问题或者获取用户输入。对话框可以由表 10.4 中的任何函数创建，这些对话框包括 errordlg，warndlg，inputdlg，uigetfile 等。
9.
模式对话框不允许在它消失之前程序中的其它窗口被访问，而普通对话框不会防碍其它窗口被访问。
10.
标准菜单是绑定到横贯图形顶部的菜单栏的，而上下文菜单可以附着到任何 GUI 组件中。当菜单栏被鼠标单击时，标准菜单即被激活，而上下文菜单必须在其所附着的 GUI 组件中右击时才能被激活。菜单从 uimenu 组件中创建，而上下文菜单同时从 uimenu 和 uicontextmenu 组件中创建。
11.
助记键即是那些可以通过敲击键盘从而使菜单项被选中的键。键盘加速键是可以执行菜单项的 CTRL + key 组合。加速键与助记键之间的主要区别是：助记键只是在菜单打开之后使菜单项被选中，而加速键则即使菜单没有被打开，它也可以触发动作（执行菜单项）。